SUBMODULAR OPTIMIZATION FOR RISK-AWARE COORDINATION OF MULTI-ROBOT SYSTEMS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Stefan Jorgensen
June 2018

This dissertation is online at: http://purl.stanford.edu/zc988gf3189

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Marco Pavone, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Mac Schwager

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Benjamin Van Roy

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Jan Vondrak

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

Robots are often designed for dangerous environments such as severe storms, but routing algorithms rarely are. This dissertation introduces a new class of routing problems with "risky traversal" where a robot may fail when travelling between two sites. Our key insight is that many objectives in the risky traversal model satisfy a diminishing returns property known as submodularity. We develop a set of tools based on submodular optimization which lead to efficient solutions for a wide variety of problems:

1. The "Team Surviving Orienteers" (TSO) problem, where the size of the team is fixed and we seek routes which maximize the expected rewards collected at nodes, subject to survival probability constraints on each robot.

2. The "On-line TSO" problem, where observations are incorporated to update the paths on-line in a parallel fashion (in response to survival events).

3. The "Heterogeneous TSO" problem, which allows robots to have different capabilities such as sensors (affecting rewards collected), actuation (affecting ability to traverse between sites), and robustness (affecting survival probabilities).

4. The "Matroid TSO" problem, where the set of routes must satisfy an 'independence' constraint represented by a matroid, for example limits on the number of each type of robot available, traffic through regions of the environment, total risk budgets for the team, or combinations of these limits.

5. The "Risk Sensitive Coverage" problem, which is the dual to the TSO where the team must satisfy a coverage constraint (e.g., ensure that nodes are visited with specified probabilities) while using minimum resources (e.g., number of robots deployed, distance travelled, or expected number of failures).

Our algorithms are based on the approximate greedy algorithm, where we iteratively select a path which approximately maximizes the expected incremental rewards subject to some constraints. Due to the submodular structure of the problems considered in this dissertation, we can prove bounds on the suboptimality of our algorithms. The approach developed in this dissertation provides a foundational set of tools for routing large scale teams in dangerous environments while explicitly planning for robot failure.

To Anneliise and Finleif.

Acknowledgments

First and foremost, I would like to thank my advisor, Prof. Marco Pavone, who has always challenged me to push deeper into research. He has set a personal example of excellence in his work that I hope I can someday emulate. I am grateful for his friendship and encouragement, especially when my research kept falling apart.

I would never have started the Ph.D. without many mentors along the way, noteably Dr. Timothy Chung, who gave me my first hands-on experiences with robots at the Naval Postgraduate School; Prof. Roger Howe, who was amoung the first to suggest I get a Ph.D.; and Prof. Andrea Goldsmith, who encouraged me to submit my first paper. I am also grateful to Prof. Jan Vondrák, who was so kind and willing to meet and discuss research ideas. Those discussions encouraged me to finish some of the most rewarding results of this dissertation. In addition I would like to thank Prof. Benjamin Van Roy and Prof. Mac Schwager for sitting on my committee and reading this dissertation.

I am grateful to my co-authors from Northrop Grumman, Dr. Mark B. Millam and Dr. Robert H. Chen, who proposed the "risky traversal" model that distinguishes our work and supported the foundations of this work. I am also grateful to Edward Schmerling, Federico Rossi, and Sumeet Singh for their help in fleshing out some of the proofs and their encouragment when research results were non-existant.

To my colleagues at ASL, thank you for making the time fly by. I have learned from each of you and wish you the best in your future endeavors.

At a personal level, I thank God for giving me this opportunity to study what I love. I am grateful to my wife Anneliise for her willingness to support me through the many ups and downs of research, to my son Finleif for giving me a *very* good reason to graduate on time, and to my parents for encouraging me throughout the Ph.D.

Finally I want to acknowledge the support of the National Science Foundation Graduate Research Fellowship Program (grant number DGE44), which gave me the means and confidence to pursue the Ph.D.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis presents a submodular optimization perspective on risk-aware coordination of multi-robot systems in dangerous environments, which leads to relatively efficient solutions for three challenging classes of routing problems. This enables large, heterogeneous teams of robots to effectively split their resources while taking calculated risks in a provably near-optimal manner. Applications include resource delivery, search and rescue missions, and environmental monitoring.

As a motivating example, consider the problem of delivering humanitarian aid in a disaster or war zone with a team of robots. There are a number of sites which need the resources, but traveling among these sites is dangerous. While the aid agency wants to deliver aid to every city, it also seeks to limit the number of assets that are lost. The goal is to maximize the *expected* number of sites visited by at least one of the vehicles, while keeping the return *probability* for each vehicle above a specified survival threshold (i.e., while fulfilling a chance constraint for the survival of each vehicle). We call this problem formulation the "Team Surviving Orienteers" (TSO) problem, illustrated in Figure 1.1. The TSO problem builds on previous work in robotics, vehicle routing, and orienteering problems by considering *risky traversal:* when a robot traverses an edge, there is a probability that it is lost and does not visit any other nodes. This creates a complex, history-dependent coupling between the edges chosen and the distribution of nodes visited, which precludes the application of existing approaches available for solving the traditional orienteering problem.

This thesis provides techniques which account for risky traversal while maintaining guarantees on solution quality and complexity. These techniques are applied to three broad problem classes: The TSO (along with variants for on-line updates and heterogeneous teams), the Matroid TSO problem (where paths must satisfy a notion of "independence" found in many applications) and the Risk Sensitive Coverage problem (which is the dual to the TSO, minimizing expected losses subject to visit probability constraints). For each problem class we provide algorithms with bounded suboptimality and linear scaling in the team size.

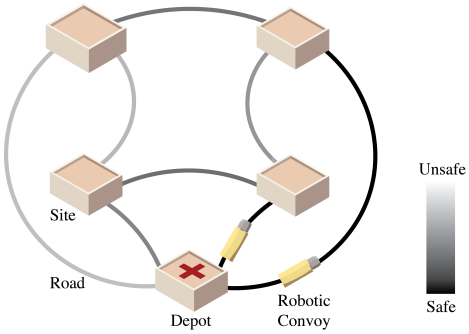


Figure 1.1: Illustration of the TSO problem applied to an aid delivery scenario. The objective is to maximize the expected number of sites visited by at least one robotic convoy. Travel between sites is risky (as emphasized by the gray color scale for each edge), and paths must be planned to ensure that the return probability for each vehicle is above a survival threshold.

1.1 Maximum Weight Routing

The problems considered in this thesis are part of a broad class of "maximum weight routing problems" which have been studied extensively by combinatorial optimization, operations research, and robotics researchers (to name a few).

In its simplest form, a maximum weight routing problem is defined by a set of nodes and edges (referred to as a graph), edge costs, a budget, and node rewards. The goal is to maximize the value of nodes visited while ensuring that the cost of edges selected remains below a budget (this is known as the "orienteering problem"). The orienteering problem has been extensively studied from the perspective of combinatorial optimization [4, 5] and is known to be NP-hard. There are many variants of the orienteering problem, such as diminishing node rewards [6, 7], team orienteering problems where many paths are planned [8], and stochastic edge costs with deadlines on reward [9].

Many variants for the OP have polynomial time approximation schema (PTAS), which is a class of algorithms that run in polynomial time and are guaranteed to return a result which has value within a constant factor of the optimum. When applied to the orienteering problem these algorithms are often constructed by calling PTAS for other sub-problems, such as spanning trees and set covers. This means that though the algorithms exist, they are often cumbersome to implement. While PTAS are useful when understanding the difficulty of a problem, the polynomial run times are typically too high order to be practical (a notable exception is [10]).

Operations research has developed the *vehicle routing problem* (VRP) [11, 12], which is a family of problems focused on finding a set of paths which maximize quality of service subject to budget or time constraints. The VRP also has many variants, mainly focused on realistic models of delivery problems in urban environments. Vehicle routing problems are often formulated as a mixed integer linear program (MILP), which is an optimization framework where all objectives and constraints are linear functions of

the problem variables, and some variables are constrained to be integral. MILP formulations are attractive because they are easy to customize and modify, which is evident from the large variety of VRP formulations in the literature. Though solving MILPs exactly is quite difficult, there are effective solvers (e.g. [13, 14]) which provide a practical solution approach to these problems.

The *Informative path planning* (IPP) problem seeks a set of paths for mobile robotic sensors in order to maximize the information gained about an environment. Many variants focus on the single-robot problem and how to incorporate challenging issues such as correlated observations (which introduces path dependence) and how to model unknown environments. One of the earliest IPP approaches [15] extends the recursive greedy algorithm of [6] using a spatial decomposition to generate paths for multiple robots. They use an "iterative assignment" technique which computes plans for each path in sequence, conditioning the rewards on the paths already planned. Since information is submodular, this greedy approach has bounded suboptimality. The structure of the IPP is most similar to the problems studied in this thesis (since it is a multi-robot path planning problem with a submodular objective function which is non-linear and history dependent), but the IPP does not capture the notion of risky traversal. Our general approach is inspired by works such as [16].

1.2 Contributions of the Thesis

The objective of this thesis is to develop approximation algorithms for the TSO, HTSO, MTSO, and RSC problems, which model fundamental challenges in risk-aware coordination of multi-robot systems. We leverage submodular optimization in order to develop our solution algorithms. We demonstrate how to linearize the risky traversal model in order to approximate the single-robot planning problem as a standard orienteering problem. The power of this approach is that our algorithms can easily be used as a PTAS, implemented as a MILP or run quickly via heuristics by using the appropriate solver for the orienteering problem.

The contributions of each chapter are summarized below:

Chapter 2: Preliminaries. In this chapter we introduce some definitions and results in submodular optimization, graphs, and the orienteering problem. We finally give the formal "risky traversal" model which is used throughout the rest of the thesis.

Chapter 3: The Team Surviving Orienteers Problem. In this chapter we study the Team Surviving Orienteers problem, where the objective is to maximize the expected rewards a team of K robots achieves while satisfying per-robot survival probability thresholds. By considering a multi-robot (team) setting with submodular node rewards (e.g. expected number of nodes visited or information gained about node random variables), we extend the state of the art for the submodular orienteering problem, and by maximizing a submodular quality of service function (with guarantees on solution quality), we extend the state of the art in the probabilistic vehicle routing literature. From a practical standpoint, the TSO problem represents a "survivability-aware" counterpart for a wide range of multi-robot coordination problems such as vehicle routing, patrolling, and informative path planning. We next show that several

broad classes of objective functions for the TSO problem are submodular, provide a linear relaxation of the single robot TSO problem (which can be solved as a standard orienteering problem), and show that the solution to the relaxed problem provides a close approximation of the optimal solution of the single robot TSO problem. Third, we propose an approximate greedy algorithm which has polynomial complexity in the number of nodes and linear complexity in the team size, and prove that the value of the output of our algorithm is lower bounded by $(1-e^{-p_s/\lambda})$OPT, where OPT is the optimum value, p_s is the survival probability threshold, and $1/\lambda \leq 1$ is the approximation factor of an oracle routine for the solution to the orienteering problem (we note that, in practice, p_s is usually close to unity).

We then consider the on-line TSO problem, where robots update their plans as they traverse the environment. All robots report their 'survival' status and re-compute their paths to maximize the expected reward given the survival states. To ensure a notion of consistency the survival probability threshold is updated in order to keep the expected number of surviving robots (conditioned on the survival states) close to $p_s K$. We give a polynomial time algorithm which approximately solves the on-line TSO problem, and provide guarantees on the performance of our on-line algorithm in terms of the objective obtained as well as bounds on the probabilities of worst-case events.

We finally discuss how to modify our algorithms to form heterogeneous teams, with similar performance guarantees and application scenarios. Crucially, the complexity increases *linearly* in the number of robot types, while preserving very similar guarantees as given for the static TSO problem.

Chapter 4: The Matroid Team Surviving Orienteers Problem. In this chapter we consider the Matroid TSO problem, which is the same as the TSO problem except instead of limiting the team size, we require that the set of paths be in the family of independent sets of a matroid. By considering matroid constraints, we extend the state of the art for the team orienteering problem, and by considering the risky traversal model we extend the state of the art for the rich vehicle routing problem. We demonstrate how to use matroids to represent a variety of constraints such as coverage, deployment, team size limitations, sub-graph diversity constraints, team-wise risk constraints, and nested cardinality constraints. We extend the approximate greedy algorithm from Chapter 3 to the MTSO setting, and prove that the value of its output is at least $\frac{p_s}{p_s+\lambda}$OPT, where OPT is the optimum value, p_s is the per-robot survival probability threshold and $1/\lambda \leq 1$ is the approximation factor of an oracle routine for the solution to the orienteering problem (we note that, in practice p_s is close to unity). We then extend the accelerated continuous greedy algorithm [17] to the MTSO problem, which is its first application to robotics. We develop a fast implementation specific to the MTSO and show its output is at least $(1-\delta)(1-e^{-p_s/(\lambda+\delta p_s)})$, where $\delta \ll 1$ is the discretization step size. We demonstrate the effectiveness of both algorithms for complex problems by considering an environmental monitoring application and give empirical performance characterizations using a synthetic dataset. We conclude the chapter by highlighting a number of extensions of this work in detail, such as p-system constraints, linear packing constraints, and coverage variants.

Chapter 5: The Risk-sensitive Coverage Problem. In this chapter we propose the Risk-Sensitive Coverage (RSC) problem and show that it is an instance of the submodular set cover problem. By considering risky traversal, we extend the state of the art for robot coverage problems, and by considering an exponentially large ground set we extend the state of the art in the submodular set cover problem. We then provide a linear relaxation which allows us to implement a cost-benefit greedy algorithm efficiently by utilizing standard orienteering problem solvers. We provide guarantees on the ratio of the minimum cost to the cost of the solution our algorithm produces. For the special case where costs are uniform our algorithm has a p_s/λ guarantee, and for the case where it is expected losses we have a $e^{-\varepsilon}p_s/\lambda$ guarantee, where p_s is the per-robot survival probability constraint, $1/\lambda \leq 1$ is the approximation guarantee for an oracle routine which solves the orienteering problem and $\varepsilon > 0$ is a tolerance parameter. We note that our solution to the cost-benefit greedy algorithm has applications far beyond the RSC problem. We then provide a bi-criteria approximation guarantee for the RSC problem, which ensures that the size of the set output by our routine is close to the optimum for a problem with similar constraints. Specifically, our result states that the cost of our solution with L robots is no more than $\frac{1}{\alpha}(1+\log(\frac{1}{\alpha}\Delta_L))$ times the optimum solution cost for a problem satisfying at least L/K fraction of the constraints, where K is the size of the final output of our algorithm, α is a approximation guarantee for the cost-benefit sub-problem, and Δ_L is the ratio of incremental coverage gain from the first and Lth path planned. We demonstrate the quality of the paths selected by our routine by comparing them with optimal paths computed for a special case of the RSC – our solution uses at most 33% more robots than the optimum. When tested on edge-sparse graphs, our cost-benefit planning approach scales empirically with the square of the number of nodes in the graph and provides near-optimal solutions (exhaustive search scales exponentially in the number of nodes). We then apply our routine to a search and rescue scenario with 225 nodes and visit probability thresholds of 0.95. Our routine finds a set of 36 paths which satisfy the constraints with approximation ratio 9. The first 13 of these paths satisfy 80.7% of the constraints with approximation ratio 4.33.

Chapter 6: Conclusions. In this chapter we present our conclusions and directions for future research.

Chapter 2

Preliminaries

In this chapter we introduce important definitions and results in submodular optimization, graph theory and routing problems. We then define our model for "risky traversal" which is used throughout the rest of the thesis.

2.1 Submodular Optimization

2.1.1 Definitions

Submodularity is the property of 'diminishing returns' for set functions. The following definitions are summarized from [18]. Given a set $\mathscr{X}$, its possible subsets are represented by $2^{\mathscr{X}}$. For two sets X and X', the set $X' \setminus X$ contains all elements in X' but not X. The *complement* of a set X contains all elements of $\mathscr{X}$ not in X, and is denoted $X^c = \mathscr{X} \setminus X$. A set function $f : 2^{\mathscr{X}} \to \mathbb{R}$ is said to be *normalized* if $f(\emptyset) = 0$ and to be *monotone* if for every $X \subseteq X' \subseteq \mathscr{X}$, $f(X) \leq f(X')$. A set function $f : 2^{\mathscr{X}} \to \mathbb{R}$ is *submodular* if for every $X \subseteq X' \subset \mathscr{X}$, $x \in \mathscr{X} \setminus X'$, we have

$$f(X \cup \{x\}) - f(X) \geq f(X' \cup \{x\}) - f(X').$$

The quantity on the left hand side is the *discrete derivative* of f at X with respect to x, which we write as $\Delta f(x \mid X)$.

The *multilinear extension* of a submodular function f can be understood as taking the expected value of the function with respect to a random set which includes each item $x \in \mathscr{X}$ with probability $y_x \in [0,1]$. Formally, let y be a vector with $|\mathscr{X}|$ elements, with element y_x corresponding to the probability $x \in R(y)$. Then we have for $x \in \mathscr{X}$,

$$\mathbb{P}\{x \in R(y)\} = y_x,$$

and for $X \subseteq \mathscr{X}$,

$$\mathbb{P}\{R(y) = X\} = \prod_{x \in X} y_x \prod_{x' \in X^c} (1 - y_{x'}).$$

The multilinear extension of f at y is defined as the expected value of $f(R(y))$:

$$F(y) := \mathbb{E}[f(R(y))] = \sum_{X \subseteq \mathscr{X}} f(X) \prod_{x \in X} y_x \prod_{x' \in X^c} (1 - y_{x'}).$$

For optimization problems the multilinear extension can be used in a similar fashion as a linear programming relaxation, where the integer variables (i.e., whether element x is in the solution) are replaced by continuous relaxations (which are easier to optimize), and the resulting solution rounded to produce a set.

An *independence system* is a tuple of a finite ground set $\mathscr{X}$ and a downward closed family of independent sets $\mathscr{I} \subseteq 2^{\mathscr{X}}$, that is if $I' \subseteq I$ and $I \in \mathscr{I}$, then $I' \in \mathscr{I}$. A *base* is an independent set $I \in \mathscr{I}$ which is inclusion-wise maximal, that is for every $x \in \mathscr{X} \setminus I$, we have $I \cup x \notin \mathscr{I}$. A *matroid* $(\mathscr{X}, \mathscr{I})$ is an independence system for which all bases have the same size (which is called the *rank* of the matroid), hence it extends the notion of linear independence to sets. There are many equivalent characterizations of matroids which are outside of the scope of this work, we refer the interested reader to [19] for more detail.

2.1.2 Cardinality constrained maximization

A typical submodular optimization problem entails finding a set $X \subseteq \mathscr{X}$ with cardinality K that maximizes f. Finding an optimal solution, X^*, is NP-hard for arbitrary submodular functions [18]. The *greedy algorithm* constructs a set $\bar{X}_K = \{x_1, \ldots, x_K\}$ by iteratively adding an element x which maximizes the discrete derivative of f at the partial set already selected. In other words the ℓth element satisfies:

$$x_\ell \in \operatorname*{argmax}_{x \in \mathscr{X} \setminus \bar{X}_{\ell-1}} \Delta f(x \mid \bar{X}_{\ell-1}).$$

We refer to the optimization problem above as 'the greedy sub-problem' at step ℓ. A well-known theorem proven by [20] states that if f is a monotone, normalized, non-negative, and submodular set function, then $f(\bar{X}_K) \geq (1 - \frac{1}{e}) f(X^*)$. This is a powerful result, but if the set $\mathscr{X}$ is large we might only be able to approximately solve the greedy sub-problem. An α–approximate greedy algorithm constructs the set $\hat{X}_K$ by iteratively adding elements which *approximately* maximize the discrete derivative of f at the partial set already selected. In particular, for some fixed $\alpha \leq 1$ the ℓth element $\hat{x}_\ell$ satisfies:

$$\Delta f(\hat{x}_\ell \mid \hat{X}_{\ell-1}) \geq \alpha \Delta f(x \mid \hat{X}_{\ell-1}) \qquad \forall x \in \mathscr{X} \setminus \hat{X}_{\ell-1}.$$

We provide a guarantee for the α–approximate greedy algorithm analogous to the guarantee for the greedy algorithm, thereby extending Theorem 4.2 of [20]:

Theorem 1 (α–approximate greedy guarantee)**.** *Let f be a monotone, normalized, non-negative, and submodular function with discrete derivative Δf. For $\alpha \in [0,1]$ and positive integer K, the output of any α–approximate greedy algorithm with $L \geq K$ elements, $\hat{X}_L$, satisfies the following inequality:*

$$f(\hat{X}_L) \geq \left(1 - e^{-\alpha L/K}\right) \max_{X \in 2^{\mathscr{X}} : |X| = K} f(X).$$

Proof. The case where $L = K$ is a special case of Theorem 1 from [21]. To generalize to $L > K$ we extend the proof for the greedy algorithm in [18]. Let $X^* \in 2^{\mathscr{X}}$ be a set which maximizes $f(X)$ subject to the cardinality constraint $|X| = K$. For $\ell < L$, we have:

$$\begin{aligned}
f(X^*) &\leq f(X^* \cup \hat{X}_\ell) \\
&= f(\hat{X}_\ell) + \sum_{k=1}^{K} \Delta f(x_k^* \mid \hat{X}_\ell \cup \{x_1^*, \ldots, x_{k-1}^*\}) \\
&\leq f(\hat{X}_\ell) + \sum_{k=1}^{K} \Delta f(x_k^* \mid \hat{X}_\ell) \\
&\leq f(\hat{X}_\ell) + \frac{1}{\alpha} \sum_{k=1}^{K} \Delta f(\hat{x}_{\ell+1} \mid \hat{X}_\ell) \\
&\leq f(\hat{X}_\ell) + \frac{K}{\alpha} (f(\hat{X}_{\ell+1}) - f(\hat{X}_\ell)).
\end{aligned}$$

The first line follows from the monotonicity of f, the second is a telescoping sum, and the third follows from the submodularity of f. The fourth line is due to the α–approximate greedy construction of $\hat{X}_L$, and the last is because all terms in the sum are equal. Now define $\delta_\ell = f(X^*) - f(\hat{X}_\ell)$. We can re-arrange the inequality above to yield:

$$\delta_{\ell+1} \leq \left(1 - \frac{\alpha}{K}\right) \delta_\ell \leq \left(1 - \frac{\alpha}{K}\right)^{\ell+1} \delta_0.$$

Since f is non-negative, $\delta_0 \leq f(X^*)$ and using the inequality $1 - x \leq e^{-x}$ we get

$$\delta_L \leq \left(1 - \frac{\alpha}{K}\right)^{L} \delta_0 \leq \left(e^{-\alpha L/K}\right) f(X^*).$$

Now substituting $\delta_L = f(X^*) - f(\hat{X}_L)$ and rearranging:

$$f(\hat{X}_L) \geq \left(1 - e^{-\alpha L/K}\right) f(X^*).$$

□

Remark: We can generalize this theorem to the case where each x_ℓ has guarantee α_ℓ. Using the same line of reasoning as in the proof for Theorem 1, we have

$$f(\hat{X}_L) \geq \left(1 - e^{-\sum_{\ell=1}^{L} \alpha_\ell / K}\right) f(X^*).$$

2.1.3 Matroid constrained maximization

Cardinality constrained maximization is an important special case of the more general matroid constrained maximization problem. Given a matroid $(\mathscr{X}, \mathscr{I})$ and a submodular function f, this problem entails finding a set $X \in \mathscr{I}$ that maximizes f. Finding an optimal solution, X^*, is NP-hard for general submodular functions, and cannot be approximated to closer than factor $(1-e^{-1})$ in polynomial time [22].

Given a matroid $(\mathscr{X}, \mathscr{I})$ with rank K and an independent set X, the *feasible set* is the set of all items not in X which can be added to X while preserving independence:

$$\mathscr{X}_F(X, \mathscr{I}) := \{x \in \mathscr{X} \setminus X \mid X \cup \{x\} \in \mathscr{I}\}.$$

The greedy algorithm is the same as in the cardinality constrained case, except the ℓth item is selected from the feasible set $\mathscr{X}_F(\bar{X}_{\ell-1}, \mathscr{I})$ rather than $\mathscr{X} \setminus \bar{X}_{\ell-1}$, and items are chosen until no more can be added, that is $\mathscr{X}_F(\bar{X}_K, \mathscr{I}) = \emptyset$. If the function f is monotone and non-negative, the greedy algorithm will choose K items, where K is the rank of the matroid $(\mathscr{X}, \mathscr{I})$. A well-known theorem proven by [23] states that if f is a monotone, normalized, non-negative, and submodular function, then $f(\bar{X}_K) \geq \frac{1}{2} f(X^*)$. Note that this is weaker than the $(1-e^{-1})$ guarantee for the cardinality constrained case.

The α–approximate greedy algorithm was shown to have a constant factor guarantee for the matroid constraint case in Appendix B of [24]:

Theorem 2 (Approximate greedy guarantee [24])**.** *Let $(\mathscr{X}, \mathscr{I})$ be a matroid with rank K and $f : 2^{\mathscr{X}} \to \mathbb{R}^+$ a non-negative monotone submodular set function. If $\hat{X}_K$ is a set chosen by an α-approximate greedy algorithm, then*

$$f(\hat{X}_K) \geq \frac{\alpha}{\alpha+1} f(X) \qquad \forall X \in \mathscr{I}.$$

Note that while the approximate greedy algorithm is simple and relatively fast, in the best case ($\alpha = 1$) the guarantee is $1/2$ whereas it is possible to achieve $1-e^{-1} \simeq 0.63$ guarantees.

The *accelerated continuous greedy algorithm* (ACGA) recently proposed by [17] optimizes the multilinear extension using coordinate gradient ascent (see pseudocode in Figure 1) and achieves a $1-e^{-1}$ guarantee as the step size goes to zero. For practical implementations the algorithm is discretized using steps of size $\delta \ll 1$. During each step the algorithm constructs an independent set by greedily maximizing the multilinear extension (line 6). After selecting each item, the corresponding component of the vector y is incremented by δ. After $1/\delta$ independent sets are selected, the fractional solutions represented by y are rounded into an integral solution by calling the `SwapRounding` procedure from [25]. In particular, Theorem 2.1 from [25] states that if X is the output of `SwapRounding` given a vector y in a matroid polytope (which always holds in the context of this paper), then X is independent, $\mathbb{E}[f(X)] \geq F(y)$, and the probability $f(X) < (1-\varepsilon)F(y)$ is bounded above by $\exp(-\varepsilon^2 F(y)/8)$.

The ACGA enjoys strong theoretical performance: [17] guarantees that the expected output (with respect to randomness from rounding) is at least fraction $(1-e^{-1}-\delta)$ of the optimum, which matches the hardness

Algorithm 1 Pseudocode for the Accelerated Continuous Greedy Algorithm

procedure ACCELCONTINUOUSGREEDY($\mathscr{I}, F, \delta$)
  $y \leftarrow \vec{0}$
  for $i = 1, \ldots, 1/\delta$ **do**
    $X_i \leftarrow \emptyset$
    for $\ell = 1, \ldots, K$ **do**
      $x_\ell \leftarrow \operatorname{argmax}_{x \in \mathscr{X}_F(X_i, \mathscr{I})} F(y + \delta \mathbf{1}_x) - F(y)$
      $y_{x_\ell} \leftarrow y_{x_\ell} + \delta$, $X_i \leftarrow X_i \cup x_\ell$
    end for
  end for
  $\hat{X} \leftarrow$ `SwapRounding`(y)
end procedure

bounds. However the ACGA selects $1/\delta$ more items than the greedy algorithm and optimizing the multilinear extension is generally difficult.

2.1.4 Submodular set cover

Given a positive cost function, $c : \mathscr{X} \to \mathbb{R}_+$, the cost of a set $X \subseteq \mathscr{X}$ is defined as $C(X) := \sum_{x \in X} c(x)$. Given a submodular function f, the *submodular set cover problem* entails finding a set with minimum cost such that the submodular function is saturated, that is $f(X) = f(\mathscr{X})$. Finding an optimal solution, X^*, to this problem is NP-hard for general submodular functions [26]. The *cost-benefit greedy algorithm* constructs a set $\bar{X}_K = \{\bar{x}_1, \ldots, \bar{x}_K\}$ by iteratively adding an element x which maximizes the ratio of the benefit (the discrete derivative of f at the partial set already selected), to the cost of the element $c(x)$. In other words the ℓth element satisfies:

$$\bar{x}_\ell \in \operatorname*{argmax}_{x \in \mathscr{X} \setminus \bar{X}_{\ell-1}} \frac{\Delta f(x \mid \bar{X}_{\ell-1})}{c(x)}.$$

We refer to the optimization problem above as 'the cost-benefit greedy sub-problem' at step ℓ. Suppose that after K iterations the coverage constraint is satisfied, i.e. $f(\bar{X}_K) = f(\mathscr{X})$. Theorem 1(ii) given by [26] states that if f is a monotone, normalized, non-negative, and submodular set function, then the cost of $\bar{X}_K$ is close to optimal,

$$C(\bar{X}_K) \leq \left(1 + \log\left(\frac{f(\bar{X}_1)c(\bar{x}_K)}{\Delta f(\bar{x}_K \mid \bar{X}_{K-1})c(\bar{x}_1)}\right)\right) C(X^*),$$

and has matching hardness bounds [22, 26]. An α–approximate cost-benefit greedy algorithm constructs a solution $\hat{X}_K$ by iteratively adding elements which approximately maximize the ratio of benefit to cost. In particular for some fixed $\alpha \leq 1$, the ℓth element $\hat{x}_\ell$ satisfies:

$$\frac{\Delta f(\hat{x}_\ell \mid \hat{X}_{\ell-1})}{c(\hat{x}_\ell)} \geq \alpha \frac{\Delta f(x \mid \hat{X}_{\ell-1})}{c(x)} \qquad \forall x \in \mathscr{X} \setminus \hat{X}_{\ell-1}.$$

We extend Theorem 1(ii) given by [26] to the approximate setting with a minor modification to their

argument (propagating α throughout):

Theorem 3 (α–approximate cost-benefit greedy [26])**.** *Let $c : \mathscr{X} \to \mathbb{R}_+$ be a positive cost function and f be a monotone, normalized, non-negative, and submodular function with discrete derivative Δf. Then for the output of any α–approximate cost-benefit greedy algorithm with K elements, $\hat{X}_K$, we have the following inequality:*

$$C(\hat{X}_K) \leq \frac{1}{\alpha}\left(1+\log\left(\frac{\frac{1}{\alpha}f(\hat{X}_1)c(\hat{x}_K)}{\Delta f(\hat{x}_K \mid \hat{X}_{K-1})c(\hat{x}_1)}\right)\right)C(X^*).$$

Proof. This result follows immediately from the proof given by [26]. When applying proposition 3 (at the top of page 389 in [26]), one must introduce the α approximation factor. Propagating the factor throughout the rest of the proof yields the result. We note that Theorem 1(i),(iii) can similarly be extended to the approximate setting, but as the results are nearly identical for our purposes, we do not state them. □

This guarantee is strong if $\alpha \simeq 1$ (meaning that the sub-problem is solved near-optimally), and if $\Delta f(\hat{x}_K \mid \hat{X}_{K-1}) \gg 0$ (meaning that the last element has meaningful contribution to satisfying the coverage constraints). In the special case that the cost is uniform, e.g. $c(x) = c$ then the cost-benefit greedy algorithm is the same as the greedy algorithm, since we only maximize the benefit.

2.2 Routing on Graphs

2.2.1 Definitions

Let $\mathscr{G}(\mathscr{V},\mathscr{E})$ denote an undirected graph, where $\mathscr{V}$ is the node set and $\mathscr{E} \subset \mathscr{V} \times \mathscr{V}$ is the edge set. Explicitly, an edge is a pair of nodes (i, j) and represents the ability to travel between nodes i and j. If the graph is directed, then the edge is an ordered pair of nodes, and represents the ability to travel from the *source node* i to the *sink node* j. A graph is called *simple* if there is only one edge which connects any given pair of nodes, *complete* if there is an edge between each pair of nodes, and *planar* if nodes can be embedded in $\mathbb{R}^d$ in such a way that the edge weight is the distance between nodes. A path is an ordered sequence of *unique* nodes such that there is an edge between adjacent nodes. For $n \geq 0$, we denote the nth node in path ρ by $\rho(n)$ and denote the number of edges in a path by $|\rho|$. Under this notation, $\rho(|\rho|)$ is the last node in path ρ. A path induces a unique set of edges, and so we use the notation $\{e \in \rho\}$ as short-hand for the set of edges $\{(\rho(n-1),\rho(n))\}_{n=1}^{|\rho|}$.

A graph $\mathscr{G}_m(\mathscr{V}_m,\mathscr{E}_m)$ is called a sub-graph of $\mathscr{G}$ if $\mathscr{V}_m \subseteq \mathscr{V}$ and $\mathscr{E}_m \subseteq \mathscr{E}$. The sub-graph of $\mathscr{G}$ induced by $\mathscr{V}' \subseteq \mathscr{V}$ is the graph $\mathscr{G}'(\mathscr{V}',\mathscr{E}')$ where $\mathscr{E}' := \{(i, j) \in \mathscr{E} \mid i, j \in \mathscr{V}'\}$.

2.2.2 The orienteering problem

The orienteering problem [27] is a classic routing problem which asks for a path which maximizes the weighted number of nodes visited given a distance constraint. Formally, given node weights $\{v_j\}_{j\in\mathscr{V}}$, edge weights $\{\omega_e\}_{e\in\mathscr{E}}$, and budget constraint C, the orienteering problem is:

$$\begin{aligned} &\underset{\rho\in\mathscr{Z}}{\text{maximize}} \sum_{j\in\rho} v_j \\ &\text{subject to} \sum_{e\in\rho} \omega_e \leq C. \end{aligned}$$

It has many variants such as rewarding edges rather than nodes, diminishing returns reward functions, team variants, and stochastic variants, to name a few [4].

2.2.3 Solution approaches

Although the orienteering problem is NP-hard, various communities of researchers have developed solution approaches focusing on different properties of the solution - performance guarantees, on-line bounds, and runtime.

Complexity theory – From a theoretical standpoint, many variants of the orienteering problem have a polynomial-time approximation scheme (PTAS), which ensures that a solution is found in polynomial time (with respect to graph size) with reward at least $1/\lambda$ times the optimal. For undirected planar graphs, [28] gives a routine with $\lambda = 1+\varepsilon$, for undirected graphs [29] gives a routine with $\lambda = (2+\varepsilon)$, and for directed graphs [6] gives a guarantee in terms of the number of nodes. The complexity of these routines are high order polynomials - for example [28] gives a $\lambda = 1+\varepsilon$ PTAS for the planar case which runs in $O(V^{16d^{3/2}/\varepsilon})$ time, where d in this context is the dimension of the plane that nodes are embedded in. Even for $\varepsilon = 1$ and $d = 2$, this is $O(V^{46})$, which is not suitable for real-world applications.

Certifiable performance applications – Practitioners who require guarantees on the quality of the solution and more modeling flexibility can use mixed integer linear programming (MILP) formulations of the orienteering problem (e.g., [30]). Commercial and open source software packages for solving MILP problems are readily available, and return an optimality gap along with the solution. Such solvers can be configured to terminate after a set amount of time or when the ratio between the current solution and upper bound becomes greater than $1/\lambda$. Note that such a solver does not provide a priori guarantees on both quality of solution and termination time, in contrast to the PTAS mentioned above.

Time-critical applications – Finally, practitioners who require fast execution but not guarantees can use a heuristic to solve the orienteering problem. There are a number of fast, high quality heuristics with open

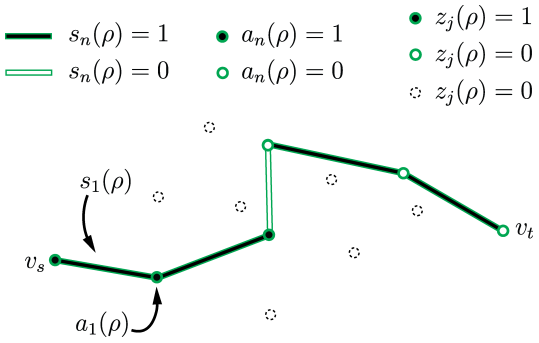


Figure 2.1: Illustration of the notation used. Robot k plans to take path ρ, whose edges are represented by lines. The fill of the lines represent the value of $s_n^k(\rho)$. In this example $s_3^k(\rho) = 0$, which means that $a_3^k(\rho) = a_4^k(\rho) = a_5^k(\rho) = 0$. The variables $z_j^k(\rho)$ are zero if either the robot fails before reaching node j or if node j is not on the planned path.

source implementations such as [31, 32]. While these heuristics do not provide guarantees, they often produce near-optimal solutions and are capable of solving large problems in seconds.

2.3 Risky Traversal Model

In this section we define the notation and setting of "Risky traversal" which we use throughout the rest of this work.

2.3.1 Definitions

Let $\mathscr{G}$ be a simple graph with $|\mathscr{V}| = V$ nodes. Edge weights $\omega : \mathscr{E} \to (0,1]$ correspond to the probability of survival for traversing an edge. Time is discretized into iterations $n = 0, 1, \ldots, N$. At iteration $n \geq 1$ a robot following path ρ traverses edge $e_\rho^n = (\rho(n-1), \rho(n))$. Robots are indexed by variable k, and for each robot and iteration we define the independent Bernoulli random variables $s_n^k(\rho)$ which are 1 with probability $\omega(e_\rho^n)$ and 0 with probability $1 - \omega(e_\rho^n)$. If robot k follows path ρ, the random variables $a_n^k(\rho) := \prod_{i=1}^n s_i^k(\rho)$ can be interpreted as being 1 if the robot 'survived' all of the edges taken until iteration n and 0 if the robot 'fails' on or before iteration n (see Figure 2.1 and Table 3.1). Given a start node v_s, a terminal node v_t, and survival probability p_s, we must find $K \geq 1$ paths $\{\rho_k\}_{k=1}^K$ (one for each of K robots) such that, for all k, the probability that $a_{|\rho_k|}^k(\rho_k) = 1$ is at least p_s, $\rho_k(0) = v_s$, and $\rho_k(|\rho_k|) = v_t$. The set of paths which satisfy these constraints is written as $\mathscr{X}(p_s, \omega)$. One can readily test whether $\mathscr{X}(p_s, \omega)$ is empty as follows: Set edge weights as $-\log(\omega(e))$, and for each node j, compute the shortest path from v_s to j, delete the edges in that path, then compute the shortest path from j to v_t. If the sum of edge weights along both paths is less than $-\log(p_s)$ then the node is reachable, otherwise it is not. Using Dijkstra's algorithm this approach can prove whether $\mathscr{X}(p_s, \omega)$ is empty after $O(V^2 \log(V))$ computations. From here on we assume that $\mathscr{X}(p_s, \omega)$ is non-empty. Define the indicator function $\mathbb{I}\{x\}$, which is 1 if x is true (or nonzero) and zero otherwise. Define

the Bernoulli random variables for $j = 1,\dots,V$, $k = 1,\dots,K$:

$$z_j^k(\rho) := \sum_{n=1}^{|\rho|} a_n^k(\rho)\,\mathbb{I}\{\rho(n) = j\},$$

which are 1 if robot k following path ρ visits node j and 0 otherwise ($z_j^k(\rho)$ is binary because a path is defined as a unique set of nodes). Note that $z_j^k(\rho)$ is independent of $z_j^{k'}(\rho')$ for $k \neq k'$, and the number of times that node j is visited by robots following the paths $\{\rho_k\}_{k=1}^K$ is given by $\sum_{k=1}^K z_j^k(\rho_k)$.

2.3.2 Poisson binomial distribution

The number of robots which visit a given node is a sum of Bernoulli random variables. The sum of K Bernoulli random variables with success probabilities $\{p_k\}_{k=1}^K$ follows the *Poisson binomial distribution*. Let $\mathscr{F}_m$ be the $\binom{K}{m}$ sets with m unique elements from $\{k\}_{k=1}^K$. For any $A \in \mathscr{F}_m$, its complement is denoted $A^c = \{k\}_{k=1}^K \setminus A$. The probability mass function for the Poisson binomial distribution is

$$f_{PB}\left(m;\{p_k\}_{k=1}^K\right) = \sum_{A\in\mathscr{F}_m} \prod_{i\in A} p_i \prod_{j\in A^c} (1-p_j),$$

which is the sum of the probabilities of each of the $\binom{K}{m}$ ways that exactly m variables are one and $K-m$ are zero. The special case where $p_k = p$ for all k, is referred to as the *binomial distribution* with parameters K and p. The binomial distribution has received much study because of its relatively simple form and extensive applications, but the Poisson binomial distribution is more difficult to analyze because each event has different probability. In the following lemma, we give a new result which gives sufficient conditions for the cumulative distribution function of a Poisson binomial random variable to be smaller than that of a specially crafted binomial random variable.

Lemma 1 (Bound for the Poisson Binomial Distribution)**.** *For $K > 2$, let f_{PB} be a Poisson binomial probability mass function with parameters $\{p_k\}_{k=1}^K$, where $p_k \leq p_K$, and let f_B be a binomial probability mass function with parameters K and $p = \frac{1}{K}\sum_{k=1}^K p_k$. Then for $M \leq (1-p_K)\left((K-2)\frac{p}{1-p}\right) + p_K$,*

$$\sum_{m=0}^{M} f_{PB}(m) \leq \sum_{m=0}^{M} f_B(m).$$

Proof. See the Appendix. □

Although one could come up with a similar bound using a binomial distribution with parameters K and p_K, it would become quite loose if K becomes large or if p_K is very close to one but p is not. Lemma 1 is less susceptible to these effects since it uses the mean of $\{p_k\}_{k=1}^K$. We use this result later to derive performance bounds for our algorithms (by setting m as the number of robots which survive to the destination), but it has much broader uses outside the context of the TSO problem.

Chapter 3

The Team Surviving Orienteers Problem

Consider the problem of delivering humanitarian aid in a disaster or war zone with a team of robots. There are a number of sites which need the resources, but traveling among these sites is dangerous. While the aid agency wants to deliver aid to every city, it also seeks to limit the number of assets that are lost. We formalize this problem as an extension of the team orienteering problem [27, 8], whereby one seeks to find a collection of paths in a doubly weighted graph which maximizes the sum of weights along all of the unique nodes in the paths while ensuring that the sum of edge weights in each path is less than a given budget. In the aid delivery case, the goal is to maximize the *expected* number of sites visited by at least one of the vehicles, while keeping the return *probability* for each vehicle above a specified survival threshold (i.e., while fulfilling a chance constraint for the survival of each vehicle). This can be seen as an extension of the team orienteering problem where edge weights are the negative log of the probability of surviving an edge, the budget is the negative log of the survival probability threshold, and node weights are the probability that the node is visited by at least one robot in the team. We call this problem formulation the "Team Surviving Orienteers" (TSO) problem, illustrated in Figure 3.1. The TSO problem builds on previous work in robotics, vehicle routing, and orienteering problems by considering *risky traversal:* when a robot traverses an edge, there is a probability that it is lost and does not visit any other nodes. This creates a complex, history-dependent coupling between the edges chosen and the distribution of nodes visited, which precludes the application of existing approaches available for solving the traditional orienteering problem.

The main goal of this chapter is to devise constant-factor approximation algorithms for the TSO problem, its extension to an on-line setting, and its extension to heterogeneous teams. Our key technical insight is that, under mild conditions, the expected number of nodes visited (or functions thereof) satisfies a diminishing returns property known as submodularity, which for set functions means that $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$. Building upon this insight, we develop a *linearization procedure* for the problem, which leads to a greedy algorithm that enjoys a constant-factor approximation guarantee. We emphasize that while a number of works have considered orienteering problems with submodular objectives [9, 6, 7] or chance constraints [33, 34] separately, the combination of the two makes the TSO problem novel, as detailed next.

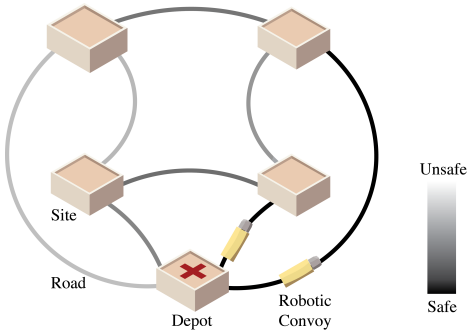


Figure 3.1: Illustration of the TSO problem applied to an aid delivery scenario. The objective is to maximize the expected number of sites visited by at least one robotic convoy. Travel between sites is risky (as emphasized by the gray color scale for each edge), and paths must be planned to ensure that the return probability for each vehicle is above a survival threshold.

The *orienteering problem* (OP) has been extensively studied from the perspective of combinatorial optimization [4, 5] and is known to be NP-hard. Over the past decade a number of constant-factor approximation algorithms have been developed for special cases of the problem [29]. Below we highlight several variants which share either similar objectives or constraints as the TSO problem.

The *submodular orienteering problem* is a generalization of the orienteering problem which considers finding a *single* path which maximizes a submodular reward function of the nodes visited. The recursive greedy algorithm proposed in [6] yields a solution in quasi-polynomial-time with reward lower bounded by $\frac{\text{OPT}}{\log(\text{OPT})}$, where OPT is the optimum value. More recently, [7] developed a (polynomial-time) generalized cost-benefit algorithm and applied it to the submodular orienteering problem. The authors show that the output of their algorithm is lower bounded by $\frac{1}{2}(1-1/e)\text{OPT}^*$, where OPT^* is the optimum given a tighter budget (i.e., $\text{OPT}^* \leq \text{OPT}$). In our context, OPT^* roughly corresponds to the maximum expected number of nodes which can be visited when the survival probability threshold is increased to $\sqrt{p_s}$. For example, if the original problem has $p_s = 0.8$, then the guarantees provided by [7] would be with respect to the maximum expected number of nodes visited when the survival probability threshold is 0.894. Depending on the node and edge weights, this may be significantly different than the optimum for the original problem, making the bound loose. Our work extends the submodular orienteering problem to the team setting for a specific class of submodular functions (i.e., the coupled node rewards and edge weights which come from risky traversal) and we provide guarantees with respect to the optimum of the original problem.

In the *orienteering problem with stochastic travel times* proposed in [9], travel times are stochastic and reward is accumulated at a node only if it is visited before a deadline. This setting could be used to solve the single robot special case of the TSO problem by using a logarithmic transformation on the survival probabilities, but [9] does not provide any polynomial-time guarantees. In the *risk-sensitive orienteering problem* [34], the goal is to maximize the sum of rewards (which is history independent) subject to a constraint on the probability that the path cost is large. The TSO problem unifies the models of the risk-sensitive and

stochastic travel time variants of the orienteering problem by considering both a submodular objective and a chance constraint on the total cost. In the TSO, we seek a *set* of paths for a team of robots which maximizes a history dependent objective function, specifically functions of the expected number of nodes visited by the team of robots. We also provide extensions for functions of multiple visits to a node, which allows broad applications such as informative path planning and property classification. Furthermore, we give an on-line version of the algorithm and provide a constant-factor guarantee for the heterogeneous team version of this problem (referred to as heterogeneous TSO –HTSO–), where robots may have different capabilities.

A second closely-related area of research is represented by the vehicle routing problem (VRP) [11, 12], which is a family of problems focused on finding a set of paths that maximize quality of service subject to budget or time constraints. The *probabilistic VRP* (PVRP) considers stochastic edge costs with chance constraints on the path costs – similar to the risk-sensitive orienteering and the TSO problem constraints. The authors of [35] pose the simultaneous location-routing problem, where both paths and depot locations are selected to minimize path costs subject to a probabilistic connectivity constraint, which specifies the average case risk rather than individual risks. More general settings were considered in [36], which considers several distribution families (such as the exponential and normal distributions), and [37], which considers non-linear risk constraints. In contrast to the TSO problem, the PVRP requires *every* node to be visited and seeks to minimize the travel cost. In the TSO problem, we require every path to be safe and maximize a function of the number of visits to each node.

A third related branch of literature is the informative path planning problem (IPP), which seeks to find a set of paths for mobile robotic sensors in order to maximize the information gained about an environment. One of the earliest IPP approaches [15] extends the recursive greedy algorithm of [6] using a spatial decomposition to generate paths for multiple robots. They use submodularity of information gain to provide performance guarantees. Sampling-based approaches to IPP were proposed by [38], which come with asymptotic guarantees on optimality. The structure of the IPP is most similar to that of the TSO problem (since it is a multi-robot path planning problem with a submodular objective function which is non-linear and history dependent), but it does not capture the notion of risky traversal which is essential to the TSO problem. Our general approach is inspired by works such as [16], but for the TSO problem we are able to further exploit the problem structure to derive constant-factor guarantees for our polynomial-time algorithm.

The contribution of this chapter is sevenfold. First, we propose the Team Surviving Orienteers (TSO) problem. By considering a multi-robot (team) setting with submodular node rewards (e.g. expected number of nodes visited or information gained about node random variables), we extend the state of the art for the submodular orienteering problem, and by maximizing a submodular quality of service function (with guarantees on solution quality), we extend the state of the art in the probabilistic vehicle routing literature. From a practical standpoint, as discussed in Section 3.1.1, the TSO problem represents a "survivability-aware" counterpart for a wide range of multi-robot coordination problems such as vehicle routing, patrolling, and informative path planning. Second, we show that several broad classes of objective functions for the TSO problem are submodular, provide a linear relaxation of the single robot TSO problem (which can be solved

Variable	Description
e_ρ^n	The nth edge in path ρ, from node $\rho(n-1)$ to $\rho(n)$
$\omega(e)$	Probability robot survives edge e
p_s	Survival threshold for each robot
$s_n^k(\rho)$	One if robot k following path ρ survives edge e_ρ^n
$a_n^k(\rho)$	One if robot k following path ρ survives to iteration n
$z_j^k(\rho)$	One if robot k following path ρ visits node j
$p_j(m, X_K)$	Probability that m of the K robots following paths in set X_K visit node j

Table 3.1: Summary of notation for the TSO problem.

as a standard orienteering problem), and show that the solution to the relaxed problem provides a close approximation of the optimal solution of the single robot TSO problem. Third, we propose an approximate greedy algorithm which has polynomial complexity in the number of nodes and linear complexity in the team size, and prove that the value of the output of our algorithm is lower bounded by $(1-e^{-p_s/\lambda})$OPT, where OPT is the optimum value, p_s is the survival probability threshold, and $1/\lambda \leq 1$ is the approximation factor of an oracle routine for the solution to the orienteering problem (we note that, in practice, p_s is usually close to unity). Fourth, we give a two-step greedy algorithm which selects two paths simultaneously and has a stronger $1-\exp(-\frac{2}{2-p_s}\frac{p_s}{\lambda})$ guarantee. Fifth, we formalize an on-line version of the TSO problem which enforces the survival constraint while taking into account the survival/failure events as they happen. Sixth, we discuss how to modify our algorithm to form heterogeneous teams, with similar performance guarantees and application scenarios. Finally, we demonstrate the effectiveness of our algorithm for large problems using simulations by solving a problem with 900 nodes and 25 robots. The work in this chapter is based on the conference paper [39] and the journal paper [40].

3.1 Problem Statement

In this section we give the formal problem statement for the static TSO problem and on-line TSO problem, provide an example, and give sufficient conditions for the objective function to be submodular.

3.1.1 Static TSO problem

We use the notation and risky traversal model outlined in Section 2.3. A summary of the notation is given in Table 3.1. The number of robots which visit node j is distributed according to a Poisson binomial distribution. Given that K robots follow the paths $\{\rho_k\}_{k=1}^K$, we write the probability that exactly m robots visit node j as

$$p_j\left(m, \{\rho_k\}_{k=1}^K\right) := f_{PB}\left(m; \left\{\mathbb{E}[z_j^k(\rho_k)]\right\}_{k=1}^K\right).$$

Finally, let $h_j : \mathbb{Z}_+ \to \mathbb{R}_+$ be a function that maps the number of visits to node j to the reward accumulated

at that node. Then the TSO problem is formally stated as:

Team Surviving Orienteers (TSO) Problem: Given a graph $\mathscr{G}$, edge weights ω, survival probability threshold p_s and team size K, maximize the expected reward of the node visits:

$$\begin{aligned} \underset{\rho_1,\ldots,\rho_K}{\text{maximize}} \quad & \sum_{j=1}^{V} \mathbb{E}\left[h_j\left(\sum_{k=1}^{K} z_j^k(\rho_k) \right) \right] & \\ \text{subject to} \quad & \mathbb{P}\left\{ a^k_{|\rho_k|}(\rho_k) = 1 \right\} \geq p_s & k = 1,\ldots,K \\ & \rho_k(0) = v_s & k = 1,\ldots,K \\ & \rho_k(|\rho_k|) = v_t & k = 1,\ldots,K \end{aligned}$$

Remarks – The objective represents the expected reward obtained by the K robots by visiting the nodes of the graph. The first set of constraints enforces the survival probability, the second and third sets of constraints enforce the initial and final node constraints. In particular, the survival probability threshold p_s serves two purposes: first, it requires that, on average, $p_s K$ robots will reach node v_t safely, and second, it enforces that risk is distributed fairly (i.e., no robot fails with too high a probability).

The model we use for risky traversal assumes that the survival random variables $s_n^k(\rho)$ are independent for all n and k. This is consistent with assumptions typical of navigation in unknown environments (e.g., SLAM applications where the environment is represented by occupancy grids), and navigation in adverse environments (e.g., due to piracy [41] or storms [42]).

The TSO problem can be viewed as a set maximization problem with a cardinality constraint, where the domain of optimization is the set $\mathscr{X}$ containing K copies of *each* path in $\mathscr{X}(p_s, \omega)$. Crucially, if the objective function is a submodular function, then Theorem 1 guarantees that the greedily selected set of paths will achieve reward close to the optimum – a central result for this chapter. Sufficient conditions for submodularity will be presented in Section 3.1.4. First, we state an online version of the TSO problem and provide an illustrative example.

3.1.2 On-line TSO problem

In the static TSO problem, the paths $\{\rho_k\}_{k=1}^K$ are computed at the beginning and then followed by the robots until the last iteration, with no path updates during execution. However at iteration n the variables $\{a_n^k(\rho_k)\}_{k=1}^K$ are observed, and this knowledge could be used to update the paths in order to account for the realized robot failures. Specifically, we seek to re-plan the paths surviving robots take such that the expected number of robots which reach node v_t safely is still $p_s K$ (consistent with the initial safety threshold), and that the risk is still distributed fairly. This can be accomplished by choosing a new survival probability threshold as follows.

Define the list of surviving robots at iteration n as $U_n := \{k \in \{1,\ldots,K\} : a_n^k(\rho_k) = 1\}$. Also, for robots $k \in U_n$, let the maximum probability that robot k can reach node v_t be denoted by ψ_k. The survival probability

threshold at iteration n, denoted as p_s^n, is computed as the solution to the optimization problem:

$$\begin{aligned} \underset{p\in(0,1)}{\text{minimize}} \quad & p \\ \text{subject to} \quad & p_s K \leq \sum_{k\in U_n} \min\{p, \psi_k\}. \end{aligned}$$

Intuitively, p_s^n represent the smallest probability threshold p such that the average number of robots which reach v_t safely will be no smaller then $p_s K$, while accounting for the fact that the maximum probability with which robot k can reach node v_t is ψ_k. If the minimization problem is infeasible, this means that for any set of paths, the expected number of robots that will reach node v_t safely is smaller than $p_s K$, and so p_s^n is simply set to one. We then define the on-line TSO problem as:

> **On-line Team Surviving Orienteers Problem:** At iteration n, given a graph $\mathcal{G}$, edge weights ω, survival probability threshold p_s^n, paths $\{\rho_k^{n-1}\}_{k=1}^K$, and the list of surviving robots U_n, select new paths $\{\rho_k^n\}_{k=1}^K$ in order to maximize the cumulative rewards:

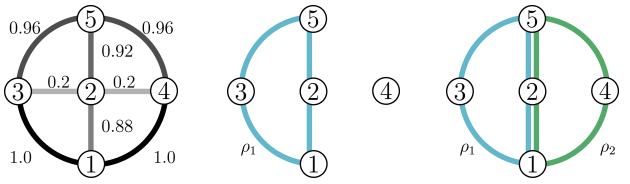
$$\begin{aligned} \underset{\rho_1^n,\ldots,\rho_K^n}{\text{maximize}} \quad & \sum_{j=1}^{V} \mathbb{E}\left[h_j\left(\sum_{k\in U_n} z_j^k(\rho_k^n) \right) \mid a_n^k(\rho_k^n) = 1, \quad k \in U_n \right] \\ \text{subject to} \quad & \rho_k^n(n') = \rho_k^{n-1}(n') \qquad n' = 1,\ldots,n, \qquad k \in U_n \\ & \rho_k^n(|\rho_k^n|) = v_t \qquad\qquad\qquad\qquad\quad k \in U_n \\ & \mathbb{P}\left\{ a_{|\rho_k^n|}^k(\rho_k^n) = 1 \right\} \geq \min\{p_s^n, \psi_k\} \qquad k \in U_n \end{aligned}$$

The objective is to maximize the expected cumulative reward conditioned on the set of surviving robots. The first constraint enforces continuity with actions taken up to iteration n, the second constraint enforces that each path ends at v_t, and the third constraint enforces the survival probability constraint. Note that if $p_s^n = 1$, this means that the number of robots which reach node v_t is expected to be less than $p_s K$ regardless of the paths chosen. If for any robot k, $p_s^n > \psi_k$, then this robot will take the one of the safest paths to v_t, and will reach v_t with probability ψ_k.

3.1.3 Example

An example of the TSO problem with a reward function that is one if the node is visited at least once and zero otherwise, is given in Figure 3.2(a). There are five nodes, and edge weights are shown next to their respective edges. Two robots start at node 1, and must end at virtual node $1'$ (which is a copy of node 1) with probability at least $p_s = 0.75$. Path $\rho_1 = \{1,3,5,2,1'\}$ is shown in Figure 3.2(b), and path $\rho_2 = \{1,4,5,2,1'\}$ is shown alongside ρ_1 in Figure 3.2(c). Robot 1 visits node 3 with probability 1.0 and node 5 with probability 0.96. Robot 2 also visits node 5 with probability 0.96 and so the probability at least one robot visits node 5 is $\mathbb{E}[1 - p_5(0, \{\rho_1, \rho_2\})] = 0.9984$. The probability that robot 1 returns safely is $\mathbb{E}\left[a_4^1(\rho_1)\right] = 0.794$. For this

simple problem, ρ_1 and ρ_2 are two of three possible paths (the third is $\{1,3,5,4,1'\}$). The expected number of nodes visited by the first robot following ρ_1 is 3.88, and for two robots following ρ_1 and ρ_2 it is 4.905. Since there are only five nodes, it is clear that adding more robots must yield diminishing returns.



(a) Graph $\mathscr{G}$ (b) Path for one robot (c) Path for two robots

Figure 3.2: (a) Example of a TSO problem. Robots start at the bottom (node 1) and darker lines correspond to safer edges. (b) A single robot can only visit four nodes safely. (c) Two robots can visit all nodes safely. It is easy to verify that adding more robots yields diminishing returns.

3.1.4 Sufficient conditions for submodular objective

The domain of optimization for the TSO problem with K robots is the set $\mathscr{X}$ that contains K copies of each element in $\mathscr{X}(p_s,\omega)$. With mild conditions on the functions $\{h_j\}_{j=1}^V$, the objective function for the TSO problem (and also for the on-line TSO problem) is submodular, as stated below.

Lemma 2 (Submodularity of the TSO problem objective)**.** *Consider a set of paths $X_K := \{\rho_k\}_{k=1}^K$ and the objective function*

$$J(X_K) = \sum_{j=1}^{V} \mathbb{E}\left[h_j\left(\sum_{k=1}^{K} z_j^k(\rho_k)\right)\right].$$

For $L \geq 1$, the objective function has discrete derivative with respect to path ρ_L at partial solution $X_{L-1} = \{\rho_\ell\}_{\ell=1}^{L-1}$,

$$\Delta J(\rho_L \mid X_{L-1}) = \sum_{j=1}^{V} \mathbb{E}\left[z_j^L(\rho_L)\right] \delta_j(X_{L-1}),$$

where we define the set function,

$$\delta_j(X_K) := \mathbb{E}\left[h_j\left(1+\sum_{k=1}^{K} z_j^k(\rho_k)\right)\right] - \mathbb{E}\left[h_j\left(\sum_{k=1}^{K} z_j^k(\rho_k)\right)\right].$$

Furthermore, the objective function is submodular if for all j, $-\delta_j(X)$ is a monotone function of X.

Proof. Let $L \geq 1$. The random variable $z_j^L(\rho_L)$ is independent of the random variables $\{z_j^\ell(\rho_\ell)\}_{\ell=1}^{L-1}$. Hence

from the definition of the discrete derivative and the tower property one has:

$$\Delta J(\rho_L \mid X_{L-1}) = \sum_{j=1}^{V} \mathbb{E}[z_j^L(\rho_L)]\mathbb{E}\left[h_j\left(1+\sum_{\ell=1}^{L-1} z_j^\ell(\rho_\ell)\right)\right] + \left(1-\mathbb{E}\left[z_j^L(\rho_L)\right]-1\right)\mathbb{E}\left[h_j\left(\sum_{\ell=1}^{L-1} z_j^\ell(\rho_\ell)\right)\right],$$

which upon simplification yields the first statement of the lemma.

We now consider the second statement of the lemma. By definition, a set function is submodular if the negative of its discrete derivative is a monotone function. If $-\delta(j)$ is monotone, then the negative of the discrete derivative is also monotone (since $\mathbb{E}[z_j^L(\cdot)] \geq 0$ and the sum of monotone functions is monotone). Hence the objective function is submodular.

□

Note that we can easily extend this result to the on-line case by conditioning on $\{a_n^k(\rho_k)\}_{k=1}^K$. For the remainder of this chapter we will restrict our attention to TSO problems that fulfill the assumptions of Lemma 2. This class of problems is indeed large; we show several examples in the next section.

3.2 Applications and Variants

The TSO problem has many applications which have submodular reward functions, which means that a greedily selected set of paths will give near-optimal rewards (as discussed in Section 2.1.2). We provide some specific examples of such applications below.

3.2.1 Aid delivery (single-visit rewards)

Consider an aid delivery problem where robots deliver a resource to sites with different demands. The reward accumulated for delivering resources to node j is the demand $d_j \geq 0$, and reward is only accumulated for the first visit. Formally, for $X_K = \{\rho_k\}_{k=1}^K$, the objective function is

$$\sum_{j=1}^{V} \mathbb{E}\left[h_j\left(\sum_{k=1}^{K} z_j^k(\rho_k)\right)\right] = \sum_{j=1}^{V} \mathbb{E}\left[d_j \mathbb{I}\left\{\sum_{k=1}^{K} z_j^k(\rho_k) > 0\right\}\right] = \sum_{j=1}^{V} d_j\,(1-p_j(0,X_K)).$$

We refer to this form of objective function as an *single-visit reward function*, because reward is only accumulated for the first visit to a node. The following lemma shows that such reward functions are submodular:

Lemma 3 (Submodularity of single-visit rewards)**.** *For $d_j \geq 0$, the single-visit reward function,*

$$\sum_{j=1}^{V} d_j(1 - p_j(0, X_K)),$$

is a normalized, non-negative, monotone, and submodular function with discrete derivative with respect to ρ_L at partial solution $X_{L-1} = \{\rho_\ell\}_{\ell=1}^{L-1}$

$$\sum_{j=1}^{V} \mathbb{E}[z_j^L(\rho_L)]\, d_j\, p_j(0, X_{L-1}).$$

Proof. Non-negativity follows from the fact that $d_j \geq 0$ and $p_j(\cdot) \geq 0$. The normalized property follows since $p_j(0, \emptyset) = 1$, and since $p_j(0, X)$ is a decreasing function of X, the objective function is monotone. For the single-visit reward function the quantity,

$$-\delta_j(X) = -d_j\, p_j(0, X),$$

is monotone, so using Lemma 2 we conclude that the objective function is submodular. □

3.2.2 Property classification (multi-visit rewards)

Now consider a multiple visit reward function where reward $h_j(m) \geq 0$ is accumulated after m visits to node j. A concrete example is a classification scenario, where each robot measures a binary property of a node imperfectly, and the objective is to minimize the posterior variance of the property distribution. If one uses a Haldane prior, which is the $\beta(0,0)$ function [43], the posterior variance after m measurements is $\frac{1}{4(m+1)}$. Setting the node priorities to $h_j(m) = \left(\frac{1}{4} - \frac{1}{4(m+1)}\right)$ gives a multi-visit reward function. Maximizing the expected cumulative rewards $h_j(m)$ is equivalent to minimizing the expected posterior variance of the distribution of the feature probabilities.

With fairly mild conditions on the rewards h_j, the multi-visit reward function is submodular, as stated in the following lemma:

Lemma 4 (Submodularity of multi-visit rewards)**.** *Let $h_j : \mathbb{Z}_+ \to \mathbb{R}_+$ be an increasing function with finite difference $\Delta_{h_j}(m) = h_j(m) - h_j(m-1)$ which satisfies the diminishing returns property*

$$\Delta_{h_j}(m+1) \leq \Delta_{h_j}(m), \qquad m \geq 1$$

and $h_j(0) = 0$. Then the reward function at the solution set $X_K = \{\rho_k\}_{k=1}^{K}$,

$$\sum_{j=1}^{V} \mathbb{E}\left[h_j\left(\sum_{k=1}^{K} z_j^k(\rho_k)\right)\right],$$

is a normalized, non-negative, monotone, and submodular function with discrete derivative with respect to ρ_L *at partial solution* $X_{L-1} = \{\rho_\ell\}_{\ell=1}^{L-1}$*:*

$$\sum_{j=1}^{V} \mathbb{E}[z_j^L(\rho_L)] \sum_{m=0}^{L-1} \Delta_{h_j}(m+1) p_j(m, X_{L-1}).$$

Proof. The reward function is non-negative because $h_j(\cdot) \geq 0$ and normalized because $h_j(0) = 0$. The number of visits to a node, $\sum_{k=1}^{K} z_j^k(\rho_k)$, is a monotone function. Since h_j is an increasing function of the number of visits to a node, this implies that the objective function is monotone. From the definition of the multi-visit reward function we have

$$\delta_j(X) = \sum_{m=0}^{|X|} \Delta_{h_j}(m+1) p_j(m, X).$$

Consider $Y = X \cup x$ and define $\gamma \in [0,1]$ such that $p_j(m,Y) = (1-\gamma) p_j(m,X) + \gamma p_j(m-1,X)$. From the definition of $\delta_j(X)$ and using the properties of h_j, we have

$$\begin{aligned}
\delta_j(X) - \delta_j(Y) &= \sum_{m=0}^{|X|} \Delta_{h_j}(m+1) p_j(m,X) \\
&\quad - \sum_{m=0}^{|X|+1} \Delta_{h_j}(m+1) \left(\gamma p_j(m-1,X) + (1-\gamma) p_j(m,X)\right) \\
&= \gamma \sum_{m=1}^{|X|} \left(\Delta_{h_j}(m+1) - \Delta_{h_j}(m+2)\right) p_j(m,X) \\
&\geq 0.
\end{aligned}$$

The first equality is derived by expressing $p_j(m,Y)$ in terms of $p_j(m,X)$, the second from simplification and the fact that $p_j(|X|+1,X) = 0$, and the inequality is due to the diminishing returns property of h_j.

This implies that $-\delta_j(X)$ is monotone and so from Lemma 2 we have that the multi-visit objective function is submodular with the stated discrete derivative. □

Note that the objective function of the feature classification example at the beginning of this subsection satisfies the conditions of Lemma 4, and hence it is a normalized, non-negative, monotone and submodular function.

3.2.3 Informative path planning

The multi-visit reward function can also model an informative path planning problem, where each node has a random variable Y_j, and the objective is to select measurements in order to minimize the entropy of the posterior distribution of $\{Y_j\}_{j=1}^{V}$ given the measurements. Setting $h_j(m)$ to be the information gained about Y_j after taking m measurements, the TSO problem becomes an informative path planning problem. It is easy to

verify that information functions satisfy the conditions of Lemma 4: the information gained from taking zero measurements is zero, information is an increasing function, and it satisfies the diminishing returns property. Hence we have that the informative path planning application of the TSO problem has a submodular objective function.

3.2.4 Variants

Edge rewards Each of the formulations above can easily be extended to a scenario where the goal is to maximize a function of the *edges* traversed. Here we describe how to extend the single-visit case, but a similar approach can be used for any of the other reward functions described above. Define $z^k_{i,j}(\rho)$ to indicate whether robot k following path ρ takes edge (i,j), and for $(i,j) \in \mathscr{E}$ define $p_{i,j}(m, \{\rho_k\}_{k=1}^K)$ as in the single visit case with $z^k_j(\cdot)$ replaced by $z^k_{i,j}(\cdot)$ (if $(i,j) \notin \mathscr{E}$, then set $p_{i,j}(0,\cdot) = 1$). Instead of node rewards d_j, we now have edge rewards $d_{i,j}$ (with $d_{i,j} = 0$ if $(i,j) \notin \mathscr{E}$), and the objective function is

$$\sum_{i=1}^{V}\sum_{j=1}^{V} d_{i,j}\,(1 - p_{i,j}(0, \{\rho_k\}_{k=1}^K)).$$

This variant could be used to model a patrolling problem, where the goal is to inspect the maximum number of roads subject to the survival probability threshold. Such problems also occur when planning scientific missions (e.g., on Mars), where the objective is to execute the most important traversals.

Visit risks Consider a scenario where the action of visiting a node is risky: a robot visiting node j survives with probability $\upsilon(j)$ and fails with probability $1 - \upsilon(j)$. We can easily incorporate this additional randomness into the TSO problem by using a directed graph to represent the traversals and directed edge weights, $\omega^d(i,j) = \omega(i,j)\upsilon(j)$, which incorporate the visit risk.

Multiple budgets We can also consider variants where there are multiple budgets, such as survival probability, distance, and time. Provided the objective is not a function of these additional budgets, our analysis extends easily.

Non-homogeneous Traversal Time The formal definitions of the static and on-line TSO problem above assume that exactly one edge is traversed per iteration by each robot. This greatly simplifies notation by allowing us to index paths with the time variable n, but is not a necessary assumption for our results.

For the static variant, homogeneous traversal time is purely a notational convenience and arbitrary times can be handled without modifying the problem statement. This is because the variables a^k_N are evaluated from the perspective of time zero and do not depend on the number of events between time zero and time N (and hence are independent of how 'time' is split).

The on-line variant is slightly more involved. At any given time t (now not necessarily the same as the index n for each path), each robot will either be at the terminal node, or be travelling to node $\rho^{t'}_k(n(\rho^{t'}_k, t))$,

where t' is the last time that robot k's path was updated, and $n(\rho_k^{t'},t)$ is defined as the index of the first node visited after time t in path $\rho_k^{t'}$. If we allow mid-course corrections (for example, with aerial vehicles), then the continuity constraints become

$$\rho_k^t(n') = \rho_k^{t'}(n') \qquad n' = 1,\ldots,n(\rho_k^{t'},t)-1, \quad k \in U_t,$$

and if we do not allow mid-course corrections (for example, on road networks), then the continuity constraints become

$$\rho_k^t(n') = \rho_k^{t'}(n') \qquad n' = 1,\ldots,n(\rho_k^{t'},t), \qquad k \in U_t.$$

In either case the fundamental structure of the problem remains the same as when traversal times are homogeneous.

3.3 Approximate Solution Approach to the Static TSO

As discussed in Section 3.1.4, we restrict our attention to TSO problems with objective functions that fulfill the assumptions of Lemma 2. Our approach to solving the TSO problem is then to exploit submodularity of the objective function using an α–approximate greedy algorithm (as defined in Section 2.1.2). In Section 3.3.1 we present a linearization of the greedy sub-problem, which in the context of the TSO problem entails finding a path which maximizes the discrete derivative of the objective function, at the partial set already constructed. We use this linearization to find a polynomial-time (p_s/λ)–approximate greedy algorithm for finding the best path given a partial solution. Leveraging this result, we describe our `GreedySurvivors` algorithm for the TSO problem in Section 3.3.2, discuss its approximation guarantee in Section 3.3.3, and characterize its computational complexity in Section 3.3.4.

3.3.1 Linear relaxation for the greedy sub-problem

Given a previously selected set of paths, $X_{L-1} = \{\rho_\ell\}_{\ell=1}^{L-1}$, the greedy sub-problem for the TSO problem at step L requires us to find a path ρ_L from the set $\mathscr{X} \setminus X_{L-1}$ which maximizes the discrete derivative of the objective function at X_{L-1} with respect to ρ_L. Note that because we define $\mathscr{X}$ to have as many copies of each path as the maximum number of robots we plan for, the set $\mathscr{X} \setminus \{\rho_\ell\}_{\ell=1}^{L-1}$ always contains at least one copy of each path in $\mathscr{X}(p_s,\omega)$. Since the discrete derivative of the objective function at X_{L-1} with respect to any of the copies of a path $\rho \in \mathscr{X}(p_s,\omega)$ is the same, we can solve the greedy sub-problem by only considering elements in the set $\mathscr{X}(p_s,\omega)$. Even with this simplification, the greedy sub-problem is very difficult for the TSO problem: it requires finding a path which maximizes submodular node rewards subject to a budget constraint (this is the submodular orienteering problem). No polynomial-time constant-factor approximation algorithm is known for general submodular orienteering problems [6], and so in this section we design one specifically for the greedy sub-problem for the TSO problem.

Under the assumptions of Lemma 2, the discrete derivative is of the form $\sum_{j=1}^{V} \mathbb{E}[z_j^L(\rho_L)]\delta_j(X_{L-1})$, for $\delta_j(X_{L-1}) \geq 0$. We relax the problem of maximizing the discrete derivative by replacing the probability that robot L traversing path ρ visits node j, $\mathbb{E}[z_j^L(\rho)]$, with the maximum probability that any robot following a feasible path can visit node j, ζ_j:

$$\zeta_j := \max_{\rho \in \mathscr{X}(p_s,\omega)} \mathbb{E}[z_j^L(\rho)].$$

For a given graph this upper bound can be found easily by using Dijkstra's algorithm with log transformed edge weights $\omega_O(e) := -\log(\omega(e))$. Let $\mathbb{I}_j(\rho)$ be equal to 1 if node j is in ρ and 0 otherwise. In the relaxed problem we are looking to maximize the sum:

$$\Delta\bar{J}(\rho \mid X_{L-1}) := \sum_{j=1}^{V} \mathbb{I}_j(\rho)\zeta_j\delta_j(X_{L-1}),$$

which represents an *optimistic* estimate of the actual discrete derivative of our objective function at X_{L-1} with respect to ρ. We can find the (approximately) best path by solving an orienteering problem as follows. Recall that for the orienteering problem we provide node weights and a constraint on the sum of edge weights (referred to as a budget), and find the path which maximizes the node rewards along the path while guaranteeing that the sum of edge weights along the path is below the budget.

We use the graph $\mathscr{G}_O$, which has the same edges and nodes as $\mathscr{G}$ but has edge weights $\omega_O(e)$ and node rewards $v_L(j) = \zeta_j\delta_j(X_{L-1})$. Solving the orienteering problem on $\mathscr{G}_O$ with budget $-\log(p_s)$ will return a path that maximizes the sum of node rewards (which is $\Delta\bar{J}(\rho \mid X_{L-1})$), and satisfies $\sum_{e\in\rho} -\log(\omega(e)) \leq -\log(p_s)$, which is equivalent to $\mathbb{P}\{a^L_{|\rho|}(\rho) = 1\} \geq p_s$.

Although solving the orienteering problem is NP-hard, there are many ways to approximate it within a constant factor $1/\lambda$ (as discussed in Section 2.2.2). Using such an oracle, we have the following guarantee:

Lemma 5 (Single robot constant-factor guarantee)**.** *Let* `Orienteering` *be a routine that solves the orienteering problem within constant-factor* $1/\lambda$*, that is for node weights* $v(j) = \zeta_j c_j$*, path* $\hat{\rho}$ *output by the routine and any path* $\rho \in \mathscr{X}(p_s, \omega)$*,*

$$\sum_{j=1}^{V} \mathbb{I}_j(\hat{\rho})v(j) \geq \frac{1}{\lambda}\sum_{j=1}^{V} \mathbb{I}_j(\rho)v(j).$$

Then for any $c_j > 0$ *and any* $\rho \in \mathscr{X}(p_s, \omega)$*, the cumulative rewards for a robot following path* $\hat{\rho}$ *satisfies*

$$\sum_{j=1}^{V} c_j\mathbb{E}[z_j(\hat{\rho})] \geq \frac{p_s}{\lambda}\sum_{j=1}^{V} c_j\mathbb{E}[z_j(\rho)].$$

Proof. By definition of ζ_j and the `Orienteering` routine, we have:

$$\sum_{j=1}^{V} c_j\mathbb{E}[z_j(\rho)] \leq \sum_{j=1}^{V} \mathbb{I}_j(\rho)\zeta_j c_j \leq \lambda\sum_{j=1}^{V} \mathbb{I}_j(\hat{\rho})\zeta_j c_j.$$

Path $\hat{\rho}$ is feasible, so $\mathbb{I}_j(\hat{\rho})p_s\zeta_j \leq \mathbb{I}_j(\hat{\rho})p_s \leq \mathbb{E}[z_j(\hat{\rho})]$, which combined with the equation above completes

the proof.

□

This is a remarkable statement because it guarantees that, if we solve the orienteering problem near-optimally, choose $c_j = \delta_j(X_{L-1})$ and p_s is not too small, the solution to the linear relaxation will give nearly the same result as the optimal solution to the greedy sub-problem at step L for the TSO problem. The intuition is that for p_s close to unity no feasible path can be very risky and so the probability that a robot *actually* reaches a node will not be too far from the maximum probability that it *could* reach the node.

3.3.2 Greedy approximation for the TSO problem

Using this relaxation with $c_j = \delta_j(X_{L-1})$ we have an p_s/λ–approximate algorithm for the greedy sub-problem at step L. This gives us a $(1-e^{-p_s/\lambda})$–approximate greedy algorithm for maximizing the discrete derivative of the objective function for the variants discussed in Section 3.2, as detailed next.

Define the method `Dijkstra`($\mathscr{G}$, i, j), which returns the length of the shortest path from i to j on the edge weighted graph $\mathscr{G}$ using Dijkstra's algorithm. Given an edge weighted graph $\mathscr{G}$ and node rewards v, the `Orienteering`$(\mathscr{G}, v)$ routine solves the orienteering problem (assuming $v_s = 1$, $v_t = V$ and budget $-\log(p_s)$) within factor $1/\lambda$, and returns the best path. Pseudocode for our algorithm is given in Algorithm 2. We begin by forming the graph $\mathscr{G}_O$ with log-transformed edge weights $\omega_O(e)$, and then use Dijkstra's algorithm to compute the maximum probability that a node can be reached. For each robot $k = 1,\ldots,K$, we solve an orienteering problem to greedily choose the path that maximizes the discrete derivative of $\bar{J}$.

Given a node index j and set of paths X, the `update`(j,X) routine returns the value of $\delta_j(X)$ as detailed below.

Algorithm 2 Pseudocode for the approximate greedy algorithm applied to the TSO problem.

procedure GREEDYSURVIVORS($\mathscr{G}, K$)
  Form $\mathscr{G}_O$ from $\mathscr{G}$, such that $v_s = 1$, $v_t = V$
  for $j = 1,\ldots,V$ **do**
    $\zeta_j \leftarrow \exp(-$`Dijkstra`$(\mathscr{G}_O, 1, j))$
  end for
  for $k = 1,\ldots,K$ **do**
    for $j = 1,\ldots,V$ **do**
      $c_j \leftarrow$ `Update`$(j, \{\rho_\ell\}_{\ell=1}^{k-1})$
      $v_k(j) \leftarrow \zeta_j c_j$
    end for
    $\rho_k \leftarrow$ `Orienteering`$(\mathscr{G}_O, v_k)$
  end for
end procedure

Updates for single-visit reward functions– Recall from Lemma 3 that for the single-visit reward function

$$\delta_j(X_L) = d_j\, p_j(0, X_L),$$

which can be computed efficiently. Initially, $\delta_j(\emptyset) = d_j$. When adding ρ_L, $\delta_j(X_L)$ updates to

$$\delta_j(X_L) \leftarrow \left(1 - \mathbb{E}[z_j^L(\rho_L)]\right) \delta_j(X_{L-1}),$$

which can be interpreted as the value of the node times the probability that none of the first L robots visit node j. The complexity of updating the node weights is $O(V)$.

Updates for multi-visit reward functions– Recall from Lemma 4 that for the multi-visit reward function

$$\delta_j(X_L) = \sum_{m=0}^{|X_L|} \Delta_{h_j}(m+1) p_j(m, X_L),$$

which can be updated by tracking the probability distribution of the number of visits to each node. The probability $p_j(m, X_L)$ can be computed recursively, since we have

$$\begin{aligned} p_j(m, X_L) = {} & \mathbb{E}[z_j^L(\rho_L)] p_j(m-1, X_{L-1}) \\ & + (1 - \mathbb{E}[z_j^L(\rho_L)]) p_j(m, X_{L-1}). \end{aligned}$$

Updating the node weights requires $O(L+1) \leq O(K)$ computations.

3.3.3 Approximation guarantees

In this section we combine the results from Section 2.1.2 and 3.3.1 to prove that the output of the `GreedySurvivors` algorithm is close to the optimal solution to the TSO problem. Specifically, we compare a team with $L \geq K$ robots using greedily selected paths to a team with K optimally selected paths, because this gives us a way to compute tighter bounds on the performance of our algorithm.

Theorem 4 (Multi-robot constant-factor guarantee)**.** *Given an* `Orienteering` *routine with constant-factor guarantee* $1/\lambda$ *as in Lemma 5, assign robot* ℓ *path* $\hat{\rho}_\ell$ *corresponding to the path returned by the* `Orienteering` *routine on graph* $\mathcal{G}_O$ *with node weights* $\nu_j = \zeta_j \delta_j(\{\hat{\rho}_k\}_{k=1}^{\ell-1})$.

Let $X_K^* = \{\rho_k^*\}_{k=1}^K$ *be an optimal solution to the TSO problem with* K *robots. For some* $L \geq K$ *and* $1 \leq \ell \leq L$*, suppose the objective is a normalized, non-negative, monotone, and submodular function with discrete derivative of the form*

$$\Delta J(\rho_\ell \mid X_{\ell-1}) = \sum_{j=1}^{V} \mathbb{E}[z_j^\ell(\rho_\ell)] \delta_j(X_{\ell-1}).$$

Then the expected cumulative reward gathered by a team of L robots with types and paths $\hat{X}_L = \{\hat{\rho}_\ell\}_{\ell=1}^{L}$ *is at least a fraction* $\gamma = \left(1 - e^{-\frac{p_s L}{\lambda K}}\right)$ *of the optimal:*

$$\sum_{j=1}^{V} \mathbb{E}\left[h_j\left(\sum_{\ell=1}^{L} z_j^\ell(\hat{\rho}_\ell)\right)\right] \geq \gamma \sum_{j=1}^{V} \mathbb{E}\left[h_j\left(\sum_{k=1}^{K} z_j^k(\rho_k^*)\right)\right].$$

Proof. The objective is a set function with domain $\mathscr{X}$, which has L copies of each feasible path. Hence for $1 \leq \ell < L$, the set $\mathscr{X} \setminus \{\hat{\rho}_k\}_{k=1}^{\ell-1}$ will always contain at least one copy of each path in $\mathscr{X}(p_s, \omega)$, and since the discrete derivative evaluated at any of the copies of the same path is the same, we can solve the greedy sub-problem by only considering elements in $\mathscr{X}(p_s, \omega)$. Using Lemma 5 with c_j chosen appropriately for the objective function, we have a constant-factor guarantee $\alpha = p_s/\lambda$ for the problem of finding the path from $\mathscr{X}(p_s, \omega)$ that maximizes the discrete derivative of our objective function. Now applying Theorem 1 to our objective function (which by assumption is normalized, non-negative, monotone, and submodular) we have the desired result. □

In many scenarios of interest p_s is quite close to 1, since robots are typically valuable or difficult to replace. For $L = K$ this theorem gives an $1 - e^{-p_s/\lambda}$ guarantee for the output of our algorithm. This bound holds for any team size, and guarantees that the output of the (polynomial-time) linearized greedy algorithm will have a similar reward to the output of the (exponential time) optimal algorithm.

Taking $L > K$ gives a practical way of testing how much more efficient the allocation for K robots could be. For example, if $L\frac{p_s}{\lambda} = 6K$ we have a $(1 - 1/e^6) \simeq 0.997$ factor approximation for the optimal value achieved by K robots. We use this approach to generate tight upper bounds for our experimental results. Note that this theorem also guarantees that as $L \to \infty$, the output of our algorithm has at least the same value as the optimum, which emphasizes the importance of guarantees for *small* teams.

Next we use the Poisson binomial bound from Section 2.3.2 to bound the probability of worst-case events, namely that a small number of robots reach node v_t safely.

Lemma 6 (Worst-case Probability Bounds)**.** *For* $K > 2$*, let* $X_K = \{\rho_k\}_{k=1}^{K}$ *be a set of paths which is a feasible solution to the TSO problem. Denote* $p_K := \max_k \mathbb{E}[z_{v_t}^k(\rho_k)]$ *and let* $\mu := \frac{1}{K}\sum_{k=1}^{K} \mathbb{E}[z_{v_t}^k(\rho_k)] \geq p_s$ *be the expected fraction of robots which will reach node* v_t*. Then for* $M \leq \lfloor (1 - p_K)(K-2)\frac{\mu}{1-\mu} + p_K \rfloor$*, the probability that M or fewer robots reach node* v_t *decreases exponentially as M decreases:*

$$\sum_{m=0}^{M} p_{v_t}(m, X_K) \leq \exp\left(-2K(\mu - M/K)^2\right).$$

Proof. Recall that if robots follow paths X_K, the probability that m robots reach node v_t is $p_{v_t}(m, X_K)$, which is the Poisson binomial probability mass function evaluated at m with parameters $\left\{\mathbb{E}[z_{v_t}^k(\rho_k)]\right\}_{k=1}^{K}$. Using Lemma 1, we have that the Poisson binomial cumulative distribution function is bounded by the binomial

cumulative distribution function with parameters K and μ. Applying Hoeffding's inequality,

$$\begin{aligned}\sum_{m=0}^{M} p_{v_t}(m, X_K) &\leq \sum_{m=0}^{M} \binom{K}{m} \mu^m (1-\mu)^{K-m} \\ &\leq \exp\left(-2\frac{(K\mu - M)^2}{K}\right)\end{aligned}$$

which after simplification is the stated result. □

This statement gives a very strong guarantee that the number of surviving robots will not be significantly below $p_s K$. For example, if $K = 25$, $\mu = 0.85$ and $p_K \leq 0.89$, then the probability that 15 or fewer robots reach v_t is less than 0.044, but the probability that 13 or fewer robots reach v_t is less than 0.0043.

3.3.4 Computational complexity

Suppose that the complexity of the `Orienteering` oracle is C_O, and the complexity of the update step is C_U. Then the complexity of our algorithm is:

$$O(V^2 \log(V)) + O(KC_U) + O(KC_O).$$

The first term is the complexity of running Dijkstra's to calculate ζ_j for all nodes, the second term is the complexity of updating the weights K times, and the final term is the complexity of solving the K orienteering problems. Generally $C_U = O(V)$ and is dominated by C_O so the asymptotic complexity of our algorithm is KC_O. Relying on an oracle routine makes the `GreedySurvivors` routine applicable for several diverse communities of researchers.

Complexity theory – From a theoretical standpoint, if a polynomial-time approximation scheme (PTAS) for the orienteering problem is used, then our algorithm is a PTAS for the TSO problem. This is a meaningful result on the complexity of the TSO problem: although the TSO is NP-hard, it can be approximated within a constant factor in polynomial time. The complexity of the best known PTAS routines for the orienteering problem and its variants are high order polynomials - for example [28] gives a $\lambda = 1 + \varepsilon$ PTAS for the planar case which runs in $O(V^{16d^{3/2}/\varepsilon})$ time, where d in this context is the dimension of the plane that nodes are embedded in. Even for $\varepsilon = 1$ and $d = 2$, this is $O(V^{46})$, which is not suitable for real-world applications.

Certifiable performance applications – Practitioners who require guarantees on the quality of the solution can use mixed integer linear programming (MILP) formulations of the orienteering problem [30]. Commercial and open source software for solving MILP problems are readily available, and return an optimality gap along with the solution. Such solvers can be configured to terminate after a set amount of time or when the ratio between the current solution and upper bound becomes greater than $1/\lambda$.

Time-critical applications – Finally, practitioners who require fast execution but not guarantees can use a heuristic to solve the orienteering problem. There are a number of fast, high quality heuristics with open source implementations such as [31, 32]. While these heuristics do not provide guarantees, they often produce near-optimal solutions and are capable of solving large problems in seconds.

3.3.5 Modifications for variants

Edge rewards – The `GreedySurvivors` routine is easily modified for the edge rewards variant. After re-defining the variables as described in Section 3.2.4, define $\zeta_{i,j} = \zeta_i \omega(i,j)$, which is the largest probability that edge (i,j) is successfully taken. The linearized greedy algorithm will still have a constant-factor guarantee, but now requires solving an *arc* orienteering problem. Constant-factor approximations for the arc orienteering problem can be found using algorithms for the OP as demonstrated in [44]: for an undirected graph $\lambda = 6+\varepsilon+o(1)$ in polynomial-time $V^{O(1/\varepsilon)}$. The arguments for Theorem 4 are the same as in the node reward case.

Walks – We can also consider *walks*, which are like a path but allow nodes and edges to be visited more than once. In this setting, $z_j^k(\rho)$ is no longer binary, and so the proofs for submodularity of the various reward functions must be updated. The argument used for Lemma 5 can be extended to walks by using an oracle which maximizes $\sum_{j=1}^{V} z_j(\rho)c_j$. If $\bar{m}$ is the maximum number of visits to a node, then this approach would give the constant factor guarantee for the greedy sub-problem as $\alpha = \frac{p_s}{\lambda \bar{m}}$. While this model does not have orienteering PTAS, it is straightforward to modify the MILP and heuristic formulations to allow for walks in in this way.

On the other hand, if we define $\mathbb{I}_j(\rho,m) := \mathbb{I}\{z_j(\rho) = m\}$, and the oracle maximizes $\sum_{j=1}^{V} \mathbb{I}_j(\rho,m)c_j(m)$, then we recover the $\frac{p_s}{\lambda}$ guarantee from Lemma 5. It is unclear whether there is an efficient MILP formulation which can act as such an oracle, though it can be posed as a Mixed Integer Program (which is generally much more difficult to solve than a MILP).

Multiple budgets – If there are multiple budgets, we can easily incorporate this into a MILP formulation by adding the appropriate path constraints. A $\lambda = (3+\varepsilon)$ PTAS for the capacitated orienteering problem was given by [45].

3.3.6 Two-step Greedy Algorithm

The `GreedySurvivors` routine above selects paths one at a time, which we call a '1-step' greedy algorithm. In this section we use a `two-step` oracle routine which selects a pair of paths, ρ_1 and ρ_2, such that for any $\rho, \rho' \in \mathscr{X}$,

$$\sum_{j \in \rho_1 \cup \rho_2} \zeta_j c_j \geq \frac{1}{\lambda} \sum_{j \in \rho \cup \rho'} \zeta_j c_j,$$

that is, the value of the nodes visited by either ρ_1 or ρ_2 is nearly as large as possible. Note that

$$\sum_{j\in\rho_1\cup\rho_2}\zeta_j c_j = \sum_{j\in\rho_1}\zeta_j c_j + \sum_{j\in\rho_2}\zeta_j c_j - \sum_{j\in\rho_1\cap\rho_2}\zeta_j c_j,$$

which can be formulated as a mixed integer quadratic objective (due to the intersection), and so we can solve the 'two-step' problem using a MIQP solver. Many of the same software packages which solve MILPs can also solve MIQPs.

For single-visit reward functions, this two-step linearization closely approximates the value of the discrete derivative $\Delta J(\rho_1,\rho_2 \mid X)$, as stated in the following lemma:

Lemma 7 (Two-step linearization)**.** *Let $J(X)$ be a single-visit reward function (as in Lemma 3) and let ρ_1, ρ_2 be the output of the* `two-step` *oracle with $c_j = d_j\, p_j(0,X)$. Then for any ρ, $\rho' \in \mathscr{X}$, we have*

$$\Delta J(\{\rho_1,\rho_2\} \mid X) \geq \frac{p_s}{(2-p_s)\lambda}\Delta J(\{\rho,\rho'\} \mid X)$$

Proof. From the definition of the single-visit reward function, we have

$$\Delta J(\{\rho,\rho'\} \mid X) = \sum_{j\in\rho}\mathbb{E}[z_j(\rho_1)]c_j + \sum_{j\in\rho_2}\mathbb{E}[z_j(\rho_2)](1-\mathbb{E}[z_j(\rho_1)])c_j$$

and so applying the definitions of ζ_j, ρ_1 and ρ_2, we get

$$\begin{aligned}
\Delta J(\{\rho,\rho'\} \mid X) &\leq \sum_{j\in\rho\cup\rho'}\zeta_j c_j + \sum_{j\in\rho\cap\rho'}\zeta_j(1-p_s)c_j \leq \sum_{j\in\rho\cup\rho'}\zeta_j(2-p_s)c_j \leq \lambda\sum_{j\in\rho_1\cup\rho_2}\zeta_j(2-p_s)c_j \\
&\leq \frac{(2-p_s)\lambda}{p_s}\left(\sum_{j\in\rho_1}\mathbb{E}[z_j(\rho_1)]c_j + \sum_{j\in\rho_2\setminus\rho_1}\mathbb{E}[z_j(\rho_2)]c_j\right) \\
&\leq \frac{(2-p_s)\lambda}{p_s}\left(\sum_{j\in\rho_1}\mathbb{E}[z_j(\rho_1)]c_j + \sum_{j\in\rho_2\setminus\rho_1}\mathbb{E}[z_j(\rho_2)]c_j + \sum_{j\in\rho_1\cap\rho_2}\mathbb{E}[z_j(\rho_2)](1-\mathbb{E}[z_j(\rho_1)])c_j\right) \\
&\leq \frac{(2-p_s)\lambda}{p_s}\Delta J(\{\rho_1,\rho_2\} \mid X)
\end{aligned}$$

□

The guarantee is weaker than in the 1-step greedy sub-problem, where the approximation factor is p_s/λ, however selecting pairs of paths has a tighter overall guarantee, as we show next:

Theorem 5 (Two-step greedy guarantee)**.** *Given a* `two-step` *routine with constant-factor guarantee $1/\lambda$, assign robots ℓ and $\ell+1$ the paths $\hat{\rho}_\ell$ and $\hat{\rho}_{\ell+1}$ returned by the* `two-step` *routine on graph $\mathscr{G}_O$ with node weights $c_j = d_j p_j(0,\hat{X}_{\ell-1})$. If $\ell = K$, then use the* `Orienteering` *routine instead of the two-step oracle.*

Let $X_K^* = \{\rho_k^*\}_{k=1}^K$ be an optimal solution to the TSO problem with K robots and let $L \geq K$ be an integer divisible by 2. For $1 \leq \ell \leq L$, suppose the objective is a normalized, non-negative, monotone, and submodular

function with discrete derivative of the form

$$\Delta J(\rho_\ell \mid X_{\ell-1}) = \sum_{j=1}^{V} \mathbb{E}[z_j^\ell(\rho_\ell)] d_j p_j(0, X_{\ell-1}).$$

Then the expected cumulative reward gathered by a team of L robots with types and paths $\hat{X}_L = \{\hat{\rho}_\ell\}_{\ell=1}^{L}$ *is at least a fraction* $\gamma = \left(1 - \exp\left(-\frac{2p_s L}{(2-p_s)\lambda K}\right)\right)$ *of the optimal.*

Proof. Following the same approach as the proof to Theorem 1, we have

$$\delta_{\ell+2} \leq \left(1 - \frac{2\alpha}{K}\right) \delta_\ell,$$

which for $\alpha = \frac{p_s}{(2-p_s)\lambda}$ gives

$$f(\hat{X}_L) \geq \left(1 - e^{-\frac{2p_s}{(2-p_s)\lambda}\frac{L}{K}}\right) f(X^*).$$

□

Remark – Note that in the ideal setting ($p_s = \lambda = 1$), this guarantees that the reward collected is at least 86% of the optimal (versus the 63% guarantee for the 1-step greedy algorithm). We can easily drop the requirement that L be divisible by two, using a similar analysis we get a guarantee of $\gamma = 1 - \exp\left(-\frac{2L-p_s}{2-p_s}\frac{p_s}{K\lambda}\right)$. Extending to multi-visit rewards is a bit more involved but should yield similar results.

3.4 Approximate Solution Approach to the On-line TSO

Information gathered on-line can be incorporated to solve the on-line TSO problem in a manner similar to the static case. There are two main structural differences between the static and the on-line planning problems: the space of feasible paths for each robot might be different (since nodes cannot be re-visited, due to the definition of paths in Section 2.2.1), and the survival constraint must be updated appropriately. These changes are handled by modifying the pre-processing step and solving a minimization problem to find p_s^n.

3.4.1 On-line algorithm

At iteration n, the on-line algorithm re-plans paths given a list of surviving robots U_n and the planned paths at the previous iteration, $X_K^{n-1} = \{\rho_k^{n-1}\}_{k=1}^{K}$. The first constraint of the on-line TSO problem requires that the first $n-1$ steps of a new plan be consistent with the past, that is $\rho_k^n(n') = \rho_k^{n-1}(n')$ for $n' \leq n-1$, which implies that the rest of the path cannot contain these nodes. We focus on finding sub-paths which do not contain any nodes already visited, start at $\rho_k^{n-1}(n)$, and end at v_t. Our algorithm consists of three stages: the first is a pre-processing stage which identifies the safest paths for every robot to reach the remaining nodes, the second stage computes the updated survival probability threshold, p_s^n, and the third stage runs the greedy algorithm to select new sub-paths.

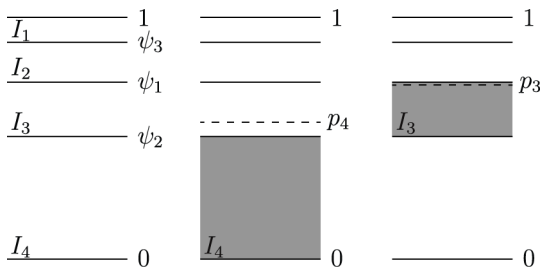


Figure 3.3: Illustration of the algorithm for updating the survival probability threshold. The maximum survival probabilities ψ_k and intervals are shown on the left. At the first step, we assume the optimum is in the interval I_4 which has the smallest upper bound (ψ_2), but this assumption is false since $p_4 > \psi_2$. At the second step we proceed to the interval with the next smallest upper bound, I_3, and find that $p_3 \in I_3$. Since the assumption is correct, we know p_3 is the optimum.

Pre-processing Due to the strict definition of paths in Section 2.2.1, robots are not permitted to re-visit nodes. Hence for each robot $k \in U_n$, we must update the maximum probability that robot k can visit each node j in $\mathscr{V}_k^n := \mathscr{V} \setminus \{\rho_k^{n-1}(n')\}_{n'=1}^{n-1}$ given that it starts from node $\rho_k^{n-1}(n)$ and cannot travel through nodes in $\{\rho_k^{n-1}(n')\}_{n'=1}^{n-1}$. We denote this probability as $\zeta_j^{k,n}$, and compute it using Dijkstra's algorithm on the graph $\mathscr{G}_k^n$ which has node set $\mathscr{V}_k^n$, edges in $\mathscr{E}$ with both the source and sink nodes in $\mathscr{V}_k^n$, and each edge given weight $-\log(\omega(e))$. The maximum probability that robot k can reach node v_t is given by $\psi_k := \zeta_{v_t}^{k,n}$.

Survival threshold update The on-line version of the TSO problem requires updating the survival probability threshold p_s^n in order to guarantee that, if possible, the risk is distributed fairly and the expected number of robots which reach node v_t safely is $p_s K$. Recall from Section 3.1.2 that p_s^n is defined as the solution to a minimization problem, and set to one if the problem is infeasible.

If $\psi_k \leq p_s^n$ for any k, this means that there is no path which satisfies the desired survival probability threshold for robot k. In this case, $\{k\}$ is removed from U_n, and ρ_k^n is set to the safest path for robot k to reach node v_t.

Solving for p_s^n is straightforward, as illustrated in Figure 3.3. The survival probability threshold p_s^n lies in one of at most $|U_n|+1$ intervals between the maximum survival probabilities $\{\psi_k\}_{k\in U_n}$. We begin by sorting the survival probabilities and guessing that the solution is in the interval with the smallest upper bound, evaluating the 'min' operator in the constraint, and then finding the value of p which makes the constraint active. If the result is in the interval we guessed, then we are done and return the result. Otherwise we move to the interval with the next smallest upper bound and repeat. If p is in none of these intervals then the problem is infeasible and we return 1. The complexity of this algorithm is $O(|U_n|(1+\log(|U_n|))) \leq O(K\log(K))$.

Greedy selection The greedy selection step is quite similar to the static TSO problem, except the survival probability threshold is now p_s^n when solving for the best path for robot k to take (the case $\psi_k \leq p_s^n$ is handled in the previous step). Because each robot has a different graph, we must solve $O(K)$ orienteering problems when selecting each path (one for each robot in U_n), which means the oracle routine is called $O(K^2)$ times

during the greedy selection step. While U_n is not empty, we set the node weights appropriately (by choosing the appropriate value for $\delta_j(X_\ell)$ conditioned on U_n and accounting for already selected paths) and find the maximum weight path with survival probability at least p_s^n for each $k \in U_n$. The most valuable path is assigned to its respective robot, that robot is removed from U_n and the loop continues.

3.4.2 Decentralized implementations

The presentation above is from a centralized perspective, where a single processing node runs all computations and sends the paths to each robot. In practice, especially for the on-line version of this problem, the robots may not be able to communicate with every other member of the team and may have noisy communications. Greedy algorithms can be decentralized by using 'iterative assignment' (e.g., as used by [16]). In this approach, each robot solves a single-robot sub-problem over its own sub-graph. A leader election is then held to determine which path has the highest discrete derivative. The winner of the election updates its plan and is removed from the pool. Remaining robots repeat the process of planning and determining a leader until every robot has a plan. If the communications graph is connected (meaning there is a way for every robot to communicate with any other robot), then this routine will yield the same result as the centralized counterpart. The communications complexity is (loosely) bounded by K^3 messages containing a path and the value of the path [46]. Finally, since each message is small (a path can be represented by $V \log_2(V)$ bits), noisy communications can be mitigated by adding strong error correction and repeated transmission.

In the case where K_d robots cannot communicate with the rest of the team, submodularity implies that the performance degrades by a factor of at most $(K - K_d)/K$. If robots not heard from are presumed 'failed', then our algorithm will make conservative choices, causing the robots to return to the terminal node sooner than if the communications were perfect. This has the added benefit that, for disk connected communications graphs, the communications network will get stronger as robots converge to v_t. This enables the list of surviving robots to be updated and the correct survival probability thresholds computed, so in a sense the communications network will be self-healing.

3.4.3 Performance guarantees

Both of the guarantees from the static TSO problem can be extended to the on-line case. The approximation guarantee can be applied because the objective function of the on-line problem inherits submodularity from the objective function of the static problem. Conditioning on U_n will change the value of the constants $\delta_k(X_{L-1})$, but not the basic form of the discrete derivative (in the sense of Lemma 2). The proofs for Lemma 5 and Theorem 4 depend only on the form of the discrete derivative, which means that we can immediately apply them by exchanging p_s with p_s^n. This means that robots following the paths output by the on-line algorithm will accumulate at least a constant factor $1 - \exp(-p_s^n/\lambda)$ of the reward accumulated by the optimal solution to the on-line TSO problem.

The on-line algorithm adapts the survival probability threshold in order to keep the expected number of

robots that *actually* reach node v_t as close to $p_s K$ as possible. When it increases p_s^n in response to robot failures, the guarantees that few robots fail become much stronger. As discussed after Lemma 6, the probability that m or fewer robots reach node v_t decreases exponentially as m decreases. So by adapting p_s^n, the on-line algorithm ensures that it is very unlikely for $|U_N|$ to be much smaller than $p_s K$.

3.4.4 Computational complexity

The computational complexity consists of four factors: pre-processing, updating the survival constraints, running the oracle, and updating the node weights. Preprocessing requires running Dijkstra's algorithm for each node and robot which has complexity $O(KV^2 \log(V))$. Updating the survival constraints requires sorting at most K elements and running at most K multiplications for the optimization routine, hence has complexity $O(K\log(K)+K)$. The oracle routine is called at most $O(K^2)$ times, since each remaining robot re-plans at every planning step. Finally, the update routine is called after each robot is selected with complexity described in Section 3.3. The total complexity is then

$$O(KV^2\log(V)) + O(K(\log(K)+1)) + O(K^2 C_O) + O(KC_U).$$

For most applications, the complexity of the oracle will dominate, and so the asymptotic complexity will typically be $O(K^2 C_O)$.

Some computation can be avoided by using the *accelerated greedy* algorithm, as discussed in [18]. The basic idea is to use the non-increasing property of the discrete derivative to quickly determine whether a given orienteering problem is worth solving. If the marginal benefit of best path for robot k at iteration n is less than the marginal benefit of some robot k' already calculated for iteration $n+1$, then we can skip re-calculating the path for robot k at iteration $n+1$. In the worst-case, this acceleration will not improve the run-time complexity, but in practice it can yield a significant improvements – in the best case, the complexity becomes $O(KC_O)$. Note that this accelerated greedy algorithm only helps when because each robot has a different feasible set, hence we cannot use it for the static algorithm.

3.5 Heterogeneous Teams

The TSO problem and our algorithm can be readily extended to a heterogeneous setting, where there are R types of robots, and we are given the co-design problem of optimizing over both paths and robot types. In Section 3.5.1 we outline the problem statement and necessary modifications to notation, in Section 3.5.2 we give sufficient conditions for the objective function to be submodular and provide an application, and in Section 3.5.3 we outline the static algorithm and guarantees for the heterogeneous case.

Variable	Description
$\omega_r(e)$	Probability robot of type r survives edge e
$p_s(r)$	Survival threshold for each type of robot
$s_n^k(r,\rho)$	One if robot k of type r following path ρ survives edge $(\rho(n-1),\rho(n))$.
$a_n^k(r,\rho)$	One if robot k of type r following path ρ survives to iteration n
$z_j^k(r,\rho)$	One if robot k of type r following path ρ visits node j
$p_j^r(m,X_K)$	Probability of m robots of type r following paths in set X_K visiting node j

Table 3.2: Summary of notation for the HTSO problem.

3.5.1 Static HTSO problem

The problem statement for the heterogeneous case is quite similar to the TSO problem, except that there are R edge weight functions and survival constraints, and any variables which previously were a function of path (e.g. s_n^k, z_j^k, and a_n^k) are now a function of path and robot type. The notation for the HTSO problem is summarized in Table 3.2 and Figure 3.4.

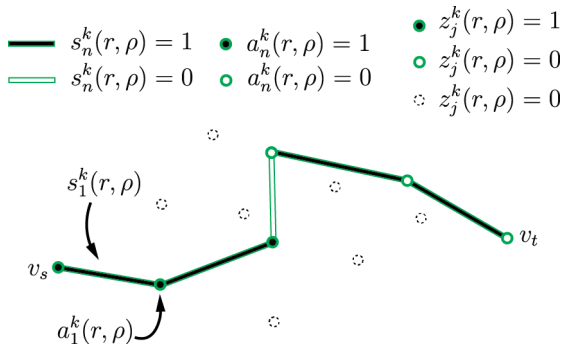


Figure 3.4: Illustration of the notation used for the HTSO (note that this is similar to Figure 2.1, except variables are now indexed by r). Robot k has type r and plans to take path ρ, whose edges are represented by lines. The fill of the lines represent the value of $s_n^k(r,\rho)$. In this example $s_3^k(r,\rho)=0$, which means that $a_3^k(r,\rho)=a_4^k(r,\rho)=a_5^k(r,\rho)=0$. The variables $z_j^k(r,\rho)$ are zero if either the robot fails before reaching node j or if node j is not on the planned path.

Given a set $X_K=\{r_k,\rho_k\}_{k=1}^K$, define the R element vector $V_j(X_K)$ element-wise as the number of robots of type r which visit node j by iteration N:

$$[V_j(X_K)]_r := \sum_{k=1}^{K} z_j^k(r,\rho_k).$$

Let the value of visiting node j with V_j visits be given by the function $H_j:\mathbb{Z}_+^R\to\mathbb{R}_+$. The HTSO problem is

defined formally as:

> **Heterogeneous Team Surviving Orienteers Problem:** Given a graph $\mathscr{G}$, edge weights ω_r, survival probability thresholds $\{p_s(r)\}_{r=1}^{R}$ and team size K, choose robot types and paths in order to maximize the expected reward accumulated by the team:

$$\begin{aligned} \underset{r_1,\rho_1,\ldots,r_K,\rho_K}{\text{maximize}} \quad & \sum_{j=1}^{V} \mathbb{E}\left[H_j\left(V_j\left(\{r_k,\rho_k\}_{k=1}^{K}\right)\right)\right] \\ \text{subject to} \quad & \mathbb{P}\{a^k_{|\rho_k|}(r_k,\rho_k)=1\} \geq p_s(r_k) \quad k=1,\ldots,K \\ & \rho_k(0)=v_s \quad k=1,\ldots,K \\ & \rho_k(|\rho_k|)=v_t \quad k=1,\ldots,K \end{aligned}$$

The objective is to choose a team of K feasible type/path pairs which maximize the expected cumulative rewards. The first constraint enforces the survival probability constraint for each path, the second and third constraints enforce that each path starts at v_s and end at v_t. Note that the reward function H_j maps a *vector* number of visits to a reward, rather than in the TSO problem, where the reward function h_j maps a scalar number of visits to a reward.

3.5.2 Submodularity and applications

We begin by characterizing when the objective function is submodular:

Lemma 8 (Submodularity of the HTSO problem objective)**.** *Let e_r be r^{th} canonical basis vector of $\mathbb{R}^R$. Given an objective function*

$$J(X_K) = \sum_{j=1}^{V} \mathbb{E}\left[H_j\left(V_j(X_K)\right)\right],$$

define the set function

$$\delta_j^r(X) = \mathbb{E}\left[H_j\left(V_j(X)+e_r\right) - H_j\left(V_j(X)\right)\right].$$

The objective function has discrete derivative with respect to (r_L,ρ_L) at partial solution $X_{L-1}=\{r_\ell,\rho_\ell\}_{\ell=1}^{L-1}$

$$\Delta J((r_L,\rho_L) \mid X_{L-1}) \sum_{j=1}^{V} \mathbb{E}\left[z_j^L(r_L,\rho_L)\right] \delta_j^{r_L}(X_{L-1}),$$

and is submodular $-\delta_j^r(X)$ is a monotone function of X for all j, r.

Proof. The random variable $z_j^L(r_L,\rho_L)$ is independent of each element of the random vector $V_j(X_{L-1})$. Hence

from the definition of the discrete derivative and the tower property we have for $x_L = (r_L, \rho_L)$

$$\begin{aligned}\Delta J(x_L \mid X_{L-1}) &= \sum_{j=1}^{V} \mathbb{E}\left[H_j\left(V_j(X_{L-1} \cup (r_L, \rho_L))\right)\right] \\ &\quad - \mathbb{E}\left[H_j\left(V_j(X_{L-1})\right)\right] \\ &= \sum_{j=1}^{V} \mathbb{E}[z_j^L(r_L, \rho_L)]\mathbb{E}\left[H_j\left(V_j(X_{L-1}) + e_{r_L}\right)\right] \\ &\quad - \left(\mathbb{E}[z_j^L(r_L, \rho_L)]\right)\mathbb{E}\left[H_j\left(V_j(X_{L-1})\right)\right],\end{aligned}$$

which upon simplification yields the first statement of the lemma. By definition, a set function is submodular if the negative of its discrete derivative is a monotone function. Since $\mathbb{E}[z_j^L(\cdot)] \geq 0$ and the sum of monotone functions is monotone, we have that the negative of the discrete derivative is monotone (hence the objective function is submodular). □

We can use Lemma 8 to immediately extend the settings described in Section 3.2 to their *uncoupled* analogues, where each robot type has its own single or multi-visit reward function, and the total reward is the sum of the rewards accumulated by each type. We can also consider *coupled* reward functions, as described next.

Consider a scenario where robot types correspond to sensor resolutions, and the information gained about a node is determined by only the highest resolution data recorded about the node. Let d_j^r the information gained about node variable j by a sensor of type r. The *best-visit reward* function is:

$$\sum_{j=1}^{V} \mathbb{E}\left[H_j(V_j(X_K))\right] = \sum_{j=1}^{V} \mathbb{E}\left[\max_k d_j^{r_k} z_j^k(r_k, \rho_k)\right]$$

Given a partial solution $X_{L-1} = \{r_\ell, \rho_\ell\}_{\ell=1}^{L-1}$, we write the probability that at least one robot of type r will visit node j as

$$p_{j,r}(X_{L-1}) = 1 - p_j^r(0, X_{L-1}),$$

and write the probability no robot of type r or less visits node j as

$$\bar{p}_{j,r}^{+}(X_{L-1}) = \prod_{r'=1}^{r} \left(1 - p_{j,r'}(X_{L-1})\right).$$

Without loss of generality, we assume that sensors with smaller type have superior resolution. The reward function depends only on the first visit for robots of a given type:

$$\sum_{j=1}^{V} \mathbb{E}\left[H_j\left(V_j(X_K)\right)\right] = \sum_{j=1}^{V} \sum_{r=1}^{R} d_j^r p_{j,r}(X_K) \bar{p}_{j,r-1}^{+}(X_K).$$

Lemma 9 (Submodularity of best-visit rewards)**.** *Let $d_j^r \geq \sum_{\hat{r}=r+1}^{R} d_j^{\hat{r}} \geq 0$. Then the reward function at the solution set $X_K = \{r_k, \rho_k\}_{k=1}^{K}$,*

$$\sum_{j=1}^{V} \mathbb{E}\left[H_j(V_j(X_K))\right] = \sum_{j=1}^{V} \mathbb{E}\left[\max_k d_j^{r_k} z_j^k(r_k, \rho_k)\right],$$

is a normalized, non-negative, monotone, and submodular function with discrete derivative with respect to (r_L, ρ_L) at partial solution $X_{L-1} = \{r_\ell, \rho_\ell\}_{\ell=1}^{L-1}$:

$$\sum_{j=1}^{V} \mathbb{E}[z_j^L(r_L, \rho_L)] \Bigg(d_j^{r_L} \bar{p}_{j,r_L}^{+}(X_{L-1}) - \sum_{r=r_L+1}^{R} d_j^r p_{j,r}(X_{L-1}) \bar{p}_{j,r-1}^{+}(X_{L-1}) \Bigg).$$

Proof. The normalized, non-negative and monotone properties follow immediately from the positivity of d_j^r, z_j^r and the fact that the maximum function is monotone. From the definition of the best-visit reward function we have

$$\delta_j^{r_L}(X) = d_j^{r_L} \bar{p}_{j,r_L}^{+}(X) - \sum_{r=r_L+1}^{R} d_j^r p_{j,r}(X) \bar{p}_{j,r-1}^{+}(X).$$

The first term can be interpreted as the negative probability that a robot of type r_L following path ρ_L is the best robot to visit the nodes in path ρ_L, and the second term is the reduction in the probability that robots in X with type $r > r_L$ will be the best type to visit nodes in path ρ_L. Consider two sets X and $Y = X \cup (\tilde{r}, \tilde{\rho})$. If $p_{j,\tilde{r}}(X) = 1$, then trivially $\delta_j^{r_L}(X) = \delta_j^{r_L}(Y)$. Otherwise, $p_{j,r}(Y) \geq p_{j,r}(X)$ with equality if $r \neq \tilde{r}$. For $r \geq \tilde{r}$, we have $\bar{p}_{j,r}^{+}(X)(1 - p_{j,\tilde{r}}(Y))/(1 - p_{j,\tilde{r}}(X)) = \bar{p}_{j,r}^{+}(Y)$, and otherwise $\bar{p}_{j,r}^{+}(X) = \bar{p}_{j,r}^{+}(Y)$. Now we show that $\delta_j^{r_L}(X) \geq \delta_j^{r_L}(Y)$ by considering three cases:

1. ($\tilde{r} \leq r_L$): From the definition of $\delta_j^{r_L}(X)$ we have

$$\delta_j^{r_L}(X) \geq \frac{1 - p_{j,\tilde{r}}(Y)}{1 - p_{j,\tilde{r}}(X)} \delta_j^{r_L}(X) = \delta_j^{r_L}(Y)$$

The inequality is due to the fact that $\frac{1-p_{j,\tilde{r}}(Y)}{1-p_{j,\tilde{r}}(X)} \leq 1$, and the equality because $r_L \geq \tilde{r}$.

2. ($\tilde{r} > r_L, p_{j,\tilde{r}}(X) = 0$): We have from the definition of $\delta_j^{r_L}(X)$:

$$\begin{aligned}
\delta_j^{r_L}(X) &= d_j^{r_L}\bar{p}_{j,r_L}^{+}(X) - \sum_{r=r_L+1}^{\tilde{r}-1} d_j^r p_{j,r}(X)\bar{p}_{j,r-1}^{+}(X) - 0 \\
&= d_j^{r_L}\bar{p}_{j,r_L}^{+}(Y) - \sum_{r=r_L+1}^{\tilde{r}-1} d_j^r p_{j,r}(Y)\bar{p}_{j,r-1}^{+}(Y) - 0 \\
&\geq d_j^{r_L}\bar{p}_{j,r_L}^{+}(Y) - \sum_{r=r_L+1}^{R} d_j^r p_{j,r}(Y)\bar{p}_{j,r-1}^{+}(Y) \\
&= \delta_j^{r_L}(Y)
\end{aligned}$$

where we use the properties introduced above for each line.

3. ($\tilde{r} > r_L, p_{j,\tilde{r}}(X) > 0$): Define $\gamma \geq 0$ such that $p_{j,\tilde{r}}(X)(1+\gamma) = p_{j,\tilde{r}}(Y)$. Then we have

$$\begin{aligned}
\delta_j^{r_L}(X) - \delta_j^{r_L}(Y) &= \gamma d_j^{\tilde{r}} p_{j,\tilde{r}}(X) p_{j,\tilde{r}-1}(X) \\
&\quad - \sum_{r=\tilde{r}+1}^{R} d_j^r p_{j,r}(X)\bar{p}_{j,r-1}^{+}(X)\left(\frac{\gamma p_{j,\tilde{r}}(X)}{1-p_{j,\tilde{r}}(X)}\right) \\
&\geq \gamma p_{j,\tilde{r}}(X)\bar{p}_{j,\tilde{r}}^{+}(X)\left(d_j^{\tilde{r}} - \sum_{r=\tilde{r}+1}^{R} d_j^r p_{j,r}(X)\right) \\
&\geq 0
\end{aligned}$$

The first and second statements are due to the definition of γ and the given identities, and the final inequality follows from the definition of $d_j^{\tilde{r}}$.

This implies that $-\delta_j^{r_L}(X)$ is a monotone function of X, which implies that the best-reward objective function is submodular.

An example which satisfies the requirement that $d_j^r \geq \sum_{\hat{r}=r+1}^{R} d_j^{\hat{r}}$ is an imaging scenario, where r corresponds to observation distance. The area covered by a picture is proportional to the square of distance, and so a small distance implies a high density of pixels (i.e. high resolution). Another example of a coupled reward function is informative path planning where each robot has a different sensor quality, and the goal is to minimize the entropy of the posterior distribution of node variables Y_j, similar to the multi-visit example from Section 3.2.

3.5.3 Algorithm

The algorithm for the static HTSO problem proceeds in an identical manner as for the TSO problem, except that at each step we must consider each of the R types of robots. We begin by computing the maximum probability that a node can be visited by a robot of type r, which we denote ζ_j^r. Then we solve R orienteering problems to find the (approximate) best type/path pair to add. Using Lemma 5 we can guarantee that each path

is within constant factor $p_s(r)/\lambda$ of the optimal path (for a fixed robot of type r), and hence the best path/pair is within constant factor $\min_r p_s(r)/\lambda$ of the optimal path/type pair for the greedy step. After choosing the path/type pair to add, we update the reward function appropriately and continue on to the next iteration.

Updates for best-visit reward functions – Recall from Lemma 9 that for the best-visit reward function

$$\delta_j^{r_L}(X_{L-1}) = d_j^{r_L}\bar{p}^+_{j,r_L}(X_{L-1}) - \sum_{r=r_L+1}^{R} d_j^r p_{j,r}(X_{L-1})\bar{p}^+_{j,r-1}(X_{L-1}),$$

which can be computed recursively by updating the visit and non-visit probabilities $p_{j,r}$ and $\bar{p}^+_{j,r}$. When (r_L, ρ_L) is added to X_{L-1}, we update the visit probabilities for $j \in \rho_L$ as

$$p_{j,r_L}(X_L) = 1 - (1 - p_{j,r_L}(X_{L-1}))(1 - \mathbb{E}[z_j^L(r_L, \rho_L)]),$$

and the non-visit probabilities for $j \in \rho_L$, $r \geq r_L$ as

$$\bar{p}^+_{j,r}(X_L) = \bar{p}^+_{j,r}(X_{L-1})(1 - \mathbb{E}[z_j^L(r_L, \rho_L)]).$$

The complexity of updating the probabilities is $O(VR)$, and updating the node weights is $O(VR^2)$. □

Guarantees We can easily get a $1 - e^{-\min_r p_s(r)/\lambda}$ constant factor guarantee by using the same approach as was used for Theorem 4. Using the remark following Theorem 1, we can provide a tighter guarantee at run-time by computing the approximation factors α_ℓ for each step of the greedy algorithm as follows. If $\hat{\rho}_\ell^r$ is the best path found for type r at step ℓ, the optimum is bounded by

$$J_\ell^{UB} = \max_r \left(1 - e^{-p_s(r)/\lambda}\right)^{-1} \sum_{j=1}^{V} \mathbb{E}\left[H_j(V_j(\hat{X}_{\ell-1} \cup \hat{\rho}_\ell^r))\right],$$

and so the approximation factor for step ℓ is bounded by the ratio of the upper bound on the optimum to the value of the approximate greedy set $\hat{X}_\ell$.

$$\alpha_\ell \geq \sum_{j=1}^{V} \mathbb{E}\left[H_j(V_j(\hat{X}_{\ell-1} \cup \rho_\ell^{r_\ell}))\right] / J_\ell^{UB},$$

which in practice will be tighter than the $1 - e^{-\min_r p_s(r)/\lambda}$ guarantee.

3.6 Numerical Experiments

In this section we provide numerous numerical experiments over a variety of synthetic and real-world graphs to characterize the performance of our algorithm in the settings described above. In Section 3.6.1 we verify that the theoretical bounds hold for a highly structured problem (where we have access to the optimal solution). In Section 3.6.2 we characterize the empirical approximation factor over a wide range of survival probability thresholds. In Sections 3.6.3 and 3.6.4 we consider real-world scenarios involving classification during a storm and information gathering in hostile environments. We demonstrate the effectiveness of simple heuristics for the orienteering problem for very large problems in Section 3.6.5. Finally we consider the on-line and heterogeneous variants in Sections 3.6.6 and 3.6.7, respectively. Unless otherwise stated, we pose the orienteering problem as a MILP and use the Gurobi solver with tolerance 10^{-4} as the oracle routine.

3.6.1 Verification of bounds

We consider a TSO problem (where we seek to maximize the expected number of nodes visited by a homogeneous team) on the graph shown in Figure 3.5(a): the central starting node has 'safe' transitions to six nodes, which have 'unsafe' transitions to the remaining twelve nodes. Due to the symmetry of the problem we can compute an optimal policy for a team of six robots, which is shown in Figure 3.5(b). The output of the greedy algorithm is shown in Figure 3.5(c). The `GreedySurvivors` solution comes close to the optimal, although the initial path planned (shown by the thick dark blue line) does not anticipate its impact on later paths. The expected number of nodes visited by robots following optimal paths, greedy paths, and the upper bound are shown in Figure 3.6. Note that the upper bound is close to the optimal, even for small teams, and that the `GreedySurvivors` performance is nearly optimal.

(a) Graph $\mathscr{G}$ (b) Optimal X_6^* (c) Greedy $\bar{X}_6$

Figure 3.5: (a) Example of a team surviving orienteers problem with depot in the center. Thick edges correspond to survival probability 0.98, light edges have survival probability 0.91. (b) Optimal paths for survival threshold $p_s = 0.70$ and $K = 6$. (c) Greedy paths for the same problem.

3.6.2 Empirical approximation factor

We compare our algorithm's performance against an upper bound on the optimal value to get a sense of the empirical versus theoretical approximation ratios. We use an exact solver for the orienteering problem, and generate instances on a complete undirected graph (meaning there is an edge between every pair of nodes) with $V = 65$ nodes and uniformly distributed edge weights in the interval $[0.3, 1)$. The upper bound used for comparison is the smallest of 1) the number of nodes which can be reached within the budget, 2) the

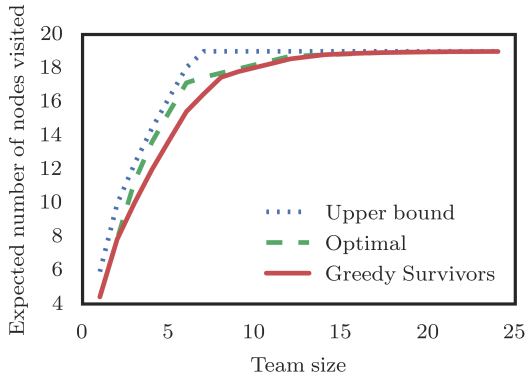


Figure 3.6: Performance comparison for the example in Figure 3.5(a). The optimal value is shown in green and the GreedySurvivors value is shown in red. The upper bound on the optimum from Theorem 4 is shown by the dotted line.

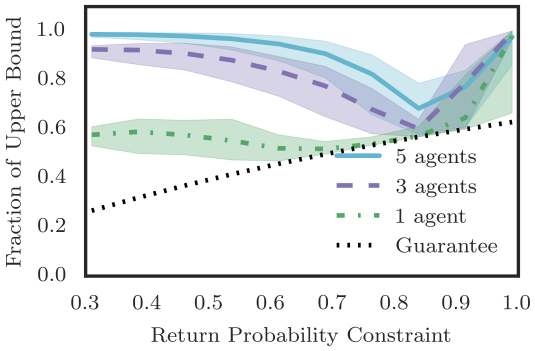


Figure 3.7: Ratio of actual result to upper bound for a 65 node complete graph. The team size ranges from 1 (at the bottom) to 5 (at the top), and in all cases a significant fraction of the possible reward is accumulated even for small p_s.

constant-factor guarantee times our approximate solution, and 3) the guarantee from solving the problem with an oversized team (from Theorem 4). The average performance (relative to the upper bound) along with the total range of results are shown in Figure 3.7, with the function $1 - e^{-p_s/\lambda}$ drawn as a dashed line. As shown, the approximation factor converges to the optimal as the team size grows. The dip around $p_s = 0.85$ is due to looseness in the bound and the fact that the optimum is not yet reached by the greedy routine.

3.6.3 Information gathering

Consider a setting where robotic sensors are used to gather information about a physical phenomena (e.g., health of coral reefs, algae blooms) in the Coral Triangle, an ecologically significant region surrounding Indonesia. Figure 3.8 shows 108 marine protected areas listed by [1]. Each area is marked by an 'X', and areas

are contained in larger regions highlighted by boxes (corresponding to relatively similar environments). One commonly proposed platform for long-duration environmental monitoring are underwater gliders [47], which have limited communication. Hence we consider the off-line TSO problem, as uncertain communications would otherwise lead to overly conservative actions (as discussed in Section 6.2).

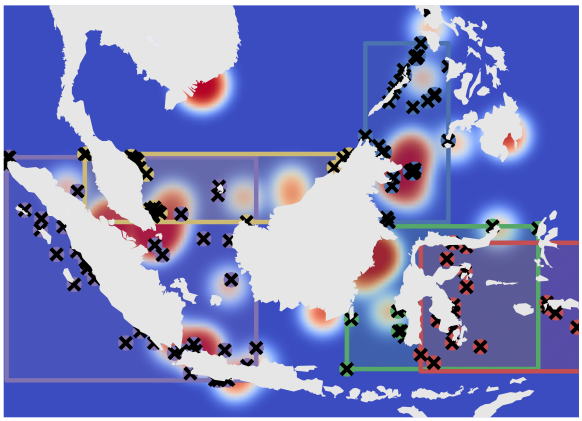


Figure 3.8: Illustration of an ocean monitoring scenario. Various regions in the Coral Triangle are outlined by boxes, sites to visit within each region are marked by 'X', and the heatmap indicates the risk of robot failure inferred from piracy incidents. Data is from the Coral Triangle Atlas [1] and IMB Piracy Reporting Centre [2].

We represent this environment using a graph with nodes corresponding to a fine uniform grid (with respect to distance). Neighboring nodes are connected (including diagonals) with edges, and we use piracy incident data [2] and a Poisson model similar to [41] to calculate the risk of traversing along the edge. Since we are not running the on-line algorithm we can simplify the problem by only considering the graph induced by the nodes corresponding to marine protected regions and edges corresponding to the shortest (maximum survival probability) path between these nodes. We assume a Gaussian measurement model, where the marginal information gain of the mth visit to node j is $\frac{1}{2}\log(1+\sigma_j^{-2}(1+m)^{-1})$, where σ_j^2 is the noise variance for measurements at node j.

Figure 3.9 compares the performance of a team of 25 robots with survival probability threshold 0.64 when using paths computed by a MILP formulation (with $\lambda = 1.5$) and using the Variable Neighborhood Search (VNS) heuristic with depth 10 (implemented as part of HeuristicLab [31]). The two approaches are compared over 30 scenarios with σ_j^2 drawn from the uniform distribution over $[0.1, 1.0]$, and objective is plotted relative to the information gained if every robot visited every node. The mean is shown as a solid line, and total range is shown using the dotted lines. The two approaches provide similar quality answers, though for these problems the VNS approach performs 8% better on average. The MILP formulation takes between 2.59 seconds and 71.01 seconds to find a path, with an average of 14.9 seconds and standard deviation of 13.6 seconds. The VNS approach takes between 17.7 and 26.8 seconds to find a path, with an average of 21.0 seconds and standard deviation of 1.52 seconds. Hence the VNS heuristic (with the given parameters)

provides a higher quality solution in a more consistent, though longer amount of time compared to the MILP approach.

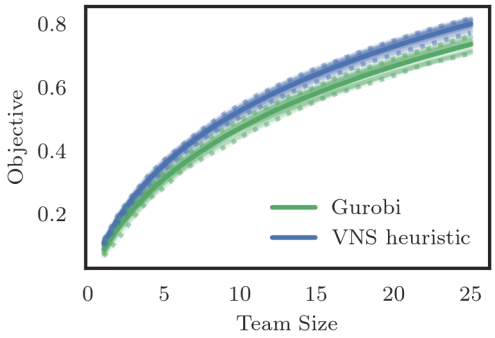


Figure 3.9: Normalized information gained by team of 25 robots when using a VNS heuristic and the MILP formulation. The mean and range are shown as solid and dotted lines, respectively. Note that depending on the parameters given, the VNS heuristic quickly produces quality solutions.

3.6.4 Classification during a storm

As the problems become large, solving MILP using personal computers becomes impractical. However cloud computing offers a low-cost way of solving even moderately large MILP problems. In this section we demonstrate the effectiveness of a 64-core cluster with 200GB of RAM in solving a problem with 225 nodes and 25 robots.

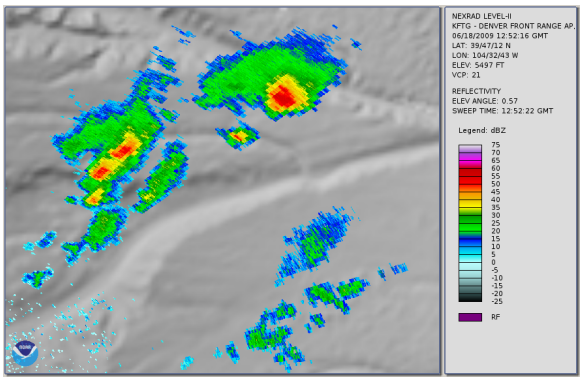


Figure 3.10: Illustration of the "base reflectivity" of a storm, which can be used to infer the danger to robots. Data from the NOAA NEXRAD level II dataset, visualization courtesy the Weather and Climate Toolkit [3].

Consider a setting where robots travel through a storm and must classify some property of each node (e.g., whether a piece of infrastructure is operational or if there are people present). We use data from the NOAA NEXRAD system [3], shown in Figure 3.10. We assume a team of relatively simple robots is

used, which in turn limits the communications available. This is because (1) wireless quality degrades with precipitation, and (2) turbulence limits the ability to steer directional antenna. Accordingly, we consider the off-line TSO problem in this setting. Robots seek to classify a single property for each node using a Haldane prior (as discussed in Section 3.2.2). Recall that this means the value for the mth visit to node j is $h_j(m) = \left(\frac{1}{4} - \frac{1}{4(m+1)}\right)$.

The risk a storm poses to robots was analyzed by [42] with a survival model very similar to ours (i.e. the product of Bernoulli random variables over each edge in a graph). For simplicity we assume that survival probability is inversely proportional to "base reflectivity" (the proportion of radar energy reflected by the weather system), though in practice one would want to use a more sophisticated approach such as the ensemble models used in [42]. We place 225 sites in a grid and compute edge weights across the straight-line

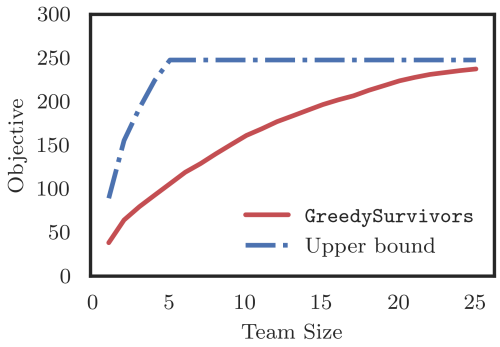


Figure 3.11: Reduction in posterior variance (using the Haldane prior) for the storm classification scenario. Note that the team of 25 robots achieves 95.8% of the maximum award available, essentially solving the problem.

connection between sites. We use a team of 25 robots with $p_s = 0.8$. Paths are found using the Gurobi solver with tolerance $\lambda = 1.5$, and each path takes an average of 55.4 seconds to find. Figure 3.11 shows the objective achieved by our team along with an upper bound (which is the smaller of the constant factor guarantee and the upper bound computed using ζ_j and k only). The team of 25 robots achieves 95.8% of the maximum reduction in posterior variance possible.

3.6.5 Large scale performance

As the problems become very large, it becomes impractical to solve them using a MILP approach. We demonstrate the usefulness of simple heuristics in solving such large problems by planning $K = 25$ paths for synthetic complete undirected graphs of various sizes. We use two `Orienteering` routines: the mixed integer formulation from [30] with Gurobi's MILP solver, and an adapted version of the open source heuristic developed by the authors of [32]. For the cases where we have comparison data (up to $V = 100$ nodes) the team using paths computed using the heuristic achieves an average of 98.2% the reward of the MILP

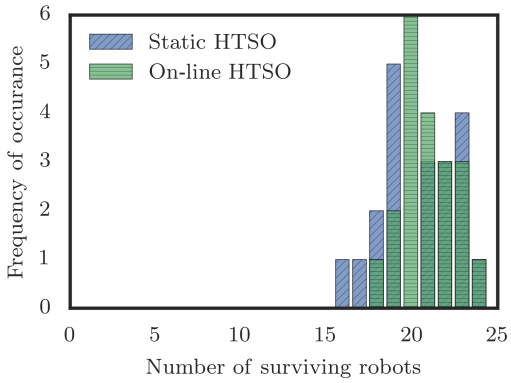


Figure 3.12: Histogram comparing surviving robots with and without re-planning for 20 trials with $K = 25$ and $p_s = 0.8$. Note that the expected number of surviving robots at the initial iteration is 20.

algorithm. Even very large problems, e.g. 25 robots on a 900 node graph, can be solved in approximately an hour with the heuristic on a machine that has a 3GHz i7 processor using 8 cores and 64GB of RAM.

3.6.6 Benefits of Re-planning

We demonstrate the benefits of an on-line approach using a synthetic complete undirected planar graph with 50 nodes and 25 homogeneous robots that have survival probability threshold $p_s = 0.8$. Figure 3.12 shows the histogram of the number of surviving robots at iteration N for 20 trials. Without re-planning, 45% of trials finished with fewer than 20 survivors, versus 15% when paths were re-planned. Conditioned on the event that the final budget is negative, the expected value of the overrun is reduced by 25% from -1.78 (without re-planning) to -1.33. A side effect of re-planning in the worst-case regime is that the average reward is reduced, but the effect is not significant. For the same set of trials discussed above, the average reward with re-planning was 93% of the reward gained without re-planning.

3.6.7 Heterogeneous team with best-visit rewards

Using heterogeneous teams can give substantial advantages over homogeneous teams of the same size. We demonstrate this by comparing teams of 20 robots on a complete undirected graph with 35 nodes in a setting where robots with higher sensor qualities have more stringent survival probability thresholds. Namely, the sensing qualities are (1, 2, 4, 8, 16), and the survival constraints are respectively (0.30, 0.45, 0.6, 0.75, 0.90). Figure 3.13 shows the expected cumulative reward as a function of the team size for the heterogeneous team (solid line) and best homogeneous teams among the five robot types (dashed line). For teams of 20 robots, the heterogeneous team vastly outperforms the homogeneous teams, in this example it is expected to achieve 186% the reward of the best homogeneous team (and 851% of the worst). We note that this is for the static HTSO problem, meaning that the paths are not re-planned based on survival events.

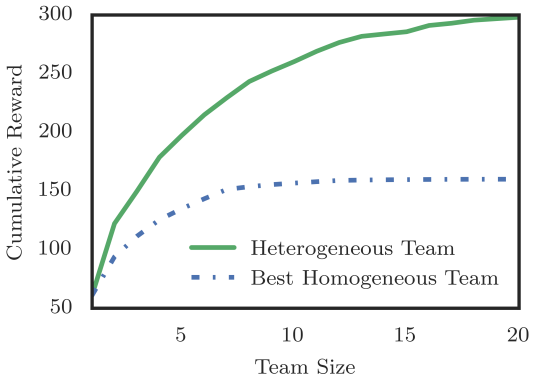


Figure 3.13: Cumulative reward of a heterogeneous team versus a homogeneous team for best-visit rewards. The graph has $V = 35$.

3.7 Conclusion

In this chapter we formulate the *Team Surviving Orienteers problem*, where we are asked to plan a set of paths that maximizes the expected cumulative reward at nodes visited while guaranteeing that every robot survives with probability at least p_s. The risky traversal model precludes the application of existing approaches available for the traditional orienteering problem. We give numerous applications where the objective function is submodular in the paths chosen, present a linearization for a class of submodular functions, and use it to develop the `GreedySurvivors` algorithm which has polynomial-time complexity with a constant-factor guarantee that the returned objective is lower bounded by $(1 - e^{-p_s/\lambda})$OPT, where OPT is the optimum.

This chapter also provides an algorithm for the on-line TSO problem, where at iteration n the list of surviving robots and edges traversed is given and we are asked to re-plan the paths in order to maximize the expected cumulative rewards, while observing an updated survival probability thresholds. We give an algorithm which finds a near-optimal set of paths which satisfy the constraints in polynomial-time.

Next we extend the TSO to the co-design problem where we are asked to assemble a heterogeneous team of robots. We give an algorithm for the HTSO with complexity that grows only linearly in the number of robots types relative to the complexity of the TSO.

Finally we demonstrate the effectiveness of our approach using numerical experiments for a variety of settings. Our experiments support the theoretical performance guarantees, and we demonstrate the efficiency of our algorithm for large graphs by solving a TSO with 25 robots and 900 nodes. We also provide scenarios which illustrate the performance gains of running the on-line version of the TSO and of using heterogeneous teams.

Chapter 4

The Matroid Team Surviving Orienteers Problem

This chapter considers the Matroid Team Surviving Orienteers (MTSO) problem, which extends the TSO problem by considering independence constraints which the set of routes must satisfy. For example, consider a scenario where mobile robotic sensors are used to monitor a number regions of the ocean, each of which may require different types of sensors. Due to weather and piracy, there is a risk that robots may "fail" when traveling from one region to another. A fleet manager seeks a set of paths which maximizes the expected number of sites monitored while satisfying various resource constraints. For example there may be limits imposed by the number of available robots of each type, the logistics of deploying the robots, the probability a given robot reaches its destination, or the amount of traffic a given region can support. If these constraints are downward closed (meaning any subset of a feasible set of paths is feasible) and satisfy an exchange property, then we can represent them using a *matroid* [19], which generalizes linear independence to set systems and has structure amenable to optimization.

We formalize this exploration problem as a generalization of the orienteering problem [27], where one seeks a path which visits as many nodes in a graph as possible given a budget constraint and travel costs. In the aforementioned example the travel costs are the probability that a robot fails while traversing between sites, and we are looking for a set of such paths which is an independent set of a matroid, maximizes the expected number of nodes visited by at least one robot and ensures that the probabilities each vehicle reaches its destination is above a specified threshold. We call this problem formulation the "Matroid Team Surviving Orienteers" (MTSO) problem, illustrated in Figure 4.1. The MTSO problem is a significant generalization of the Team Surviving Orienteers (TSO) problem presented in Chapter 3, which only imposes a maximum team size constraint (a very special case of a matroid constraint). Both the MTSO and TSO are distinct from other work because of the notion of *risky traversal:* when a robot traverses an edge, there is a probability that it fails and does not visit any other nodes. This creates a complex, path-dependent coupling between the edges

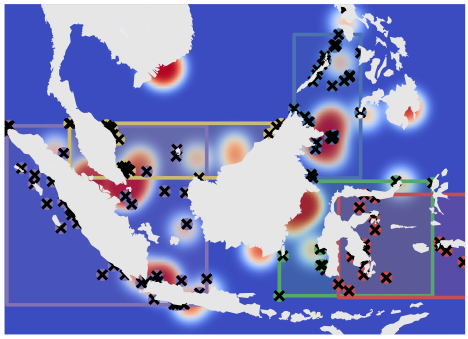


Figure 4.1: Illustration of the MTSO setting for an ocean monitoring scenario. Various regions in the Coral Triangle are outlined by boxes, sites to visit within each region are marked by 'X', and the heat map indicates the risk of robot failure inferred from piracy incidents. Data is from the Coral Triangle Atlas [1] and IMB Piracy Reporting Centre [2]. The objective is to find a set of paths for a heterogeneous team which maximizes the expected number of sites visited subject to survival probability constraints and independence constraints (e.g. limits on team size or sensor quantities).

chosen and the distribution of nodes visited, which precludes the application of existing approaches available for the traditional orienteering problem.

The objective of this chapter is to devise constant-factor approximation algorithms for the MTSO problem. Our key technical insight is that we can exploit submodularity of the objective function to design efficient solution algorithms for the much more general MTSO problem. We describe numerous applications of matroid constraints to path planning problems, and describe an approximate greedy algorithm which enjoys a constant-factor approximation guarantee. We also extend the accelerated continuous greedy algorithm [17] which has stronger guarantees and can be applied to even more general settings than the MTSO problem. Although a number of works have considered routing problems with submodular objectives [6, 9, 7], chance constraints [33, 34], or downward closed constraints [48, 49, 50] separately, the MTSO is novel because it combines all three aspects.

A closely related area of research is the *rich vehicle routing problem* (RVRP) [49], which considers settings such as routing heterogeneous teams [48], "fleet dimensioning" (choosing team composition) [50], and incompatability constraints [51]. The vast majority of solution algorithms for the RVRP are heuristic [49] and do not consider risky traversal. A notable exception are *linear temporal logical* (LTL) constraints, which have exact integer programming formulations [52] and transition system formulations [53, 54], the complexity of which grows exponentially in the team size. We consider a narrower yet still expressive set of constraints and derive polynomial time solution algorithms with constant factor approximation guarantees.

Matroids have been applied with great success to many fields of engineering such as electrical and structural network design [55], but they have rarely been used for robotic routing problems. In [56], path constraints are represented using a p-system (which generalizes a matroid), and a submodular maximization problem with p-system constraints is solved to find the approximately most informative path for a single agent. Multi-robot task allocation problems are cast as decentralized submodular maximization problems

with matroid constraints by [57] (who considers private reward functions), and [58] (who considers communication constraints). In both cases the objective is to assign robots to tasks, rather than to find a high reward set of paths for a team of robots. A partition matroid is used by [59] to model an informative path planning problem. They focus on a distributed variant of the greedy algorithm and do not consider risky traversal. We apply submodular maximization at a higher level of abstraction and use an orienteering problem oracle in order to find high quality paths for each robot. This requires different analysis which utilizes results on approximate greedy algorithms for submodular maximization subject to matroid [23] and p-system [24] independence constraints.

The contribution of this chapter is six-fold. First, we propose a generalization of the TSO problem, referred to as the Matroid TSO problem. By considering matroid constraints, we extend the state of the art for the team orienteering problem, and by considering the risky traversal model we extend the state of the art for the rich vehicle routing problem. Second, we demonstrate how to use matroids to represent a variety of constraints such as coverage, deployment, team size limitations, sub-graph diversity constraints, team-wise risk constraints, and nested cardinality constraints. Third, we extend the approximate greedy algorithm (presented in Chapter 3) to the MTSO setting, and prove that the value of its output is at least $\frac{p_s}{p_s+\lambda}$OPT, where OPT is the optimum value, p_s is the per-robot survival probability threshold and $1/\lambda \leq 1$ is the approximation factor of an oracle routine for the solution to the orienteering problem (we note that, in practice p_s is close to unity). Fourth, we extend the accelerated continuous greedy algorithm [17] to the MTSO problem, which is its first application to robotics. We develop a fast implementation specific to the MTSO and show its output is at least $(1-\delta)(1-e^{-p_s/(\lambda+\delta p_s)})$, where $\delta \ll 1$ is the discretization step size. Fifth, we demonstrate the effectiveness of both algorithms for complex problems by considering an environmental monitoring application and give an empirical timing characterization for both algorithms. Finally, we highlight a number of extensions of this work in detail, such as p-system constraints, linear packing constraints, and coverage variants. The MTSO and greedy algorithm were published in our conference chapter [60].

4.1 The Matroid TSO Problem

Given a start node v_s, a terminal node v_t, and survival probability p_s we must find at most $K \geq 1$ paths $\{\rho_k\}_{k=1}^{K}$ (one for each of K robots) such that, for all k, the probability that $a^k_{|\rho_k|}(\rho_k) = 1$ is at least p_s, $\rho_k(0) = v_s$ and $\rho_k(|\rho_k|) = v_t$. The set of paths which satisfy these constraints is written as $\mathscr{X}(p_s, \omega)$. In this chapter we are primarily interested in the probability that no robots visit node j, which has the simple expression:

Let $d_j > 0$ be the reward accumulated for visiting node j at least once, and define $\mathscr{X}$ as the set containing K copies of *each* path in $\mathscr{X}(p_s, \omega)$. Given a matroid $(\mathscr{X}, \mathscr{I})$ with rank K, we are interested in finding an independent set which maximizes the weighted expected number of nodes visited. Since the objective is non-negative and submodular (as discussed in Chapter 3), we assume without loss of generality that the size of the optimizing set is K, and state the Matroid TSO problem formally:

Matroid TSO (MTSO) Problem: Given a graph $\mathscr{G}$, edge weights ω, survival probability threshold p_s and matroid $(\mathscr{X}, \mathscr{I})$, maximize the weighted expected number of nodes visited by at least one robot:

$$\begin{aligned} \underset{\{\rho_k\}_{k=1}^K \in \mathscr{I}}{\text{maximize}} \quad & \sum_{j=1}^{V} d_j \left(1 - p_j\left(0, \{\rho_k\}_{k=1}^K\right)\right) \\ \text{subject to} \quad & \mathbb{P}\left\{a^k_{|\rho_k|}(\rho) = 1\right\} \geq p_s \quad k = 1, \ldots, K \\ & \rho_k(0) = v_s \qquad\qquad\quad k = 1, \ldots, K \\ & \rho_k(|\rho|) = v_t \qquad\qquad k = 1, \ldots, K \end{aligned}$$

The optimization is over the independent sets of the matroid, and the solution is a set of K paths. The objective represents the cumulative expected reward obtained by the robots following the selected paths. The first set of constraints enforces the survival probability, the second and third sets of constraints enforce the initial and final node constraints. Note that a single visit reward function is used here for the sake of simplicity, our approach and formulation extend easily to the multi-visit reward functions discussed in Chapter 3.

The MTSO problem can be viewed as a set maximization problem with a matroid constraint, where the domain of optimization is the family of independent sets $\mathscr{I}$. Crucially, since the objective is a submodular function, Theorem 2 (proved by [24]) implies that the greedily selected set of paths will achieve reward close to the optimum.

4.2 Applications and Variants

In this section we highlight several examples of matroids and their applications in the context of the MTSO problem.

4.2.1 Uniform matroid

Given a positive integer K, the independent sets of a uniform matroid are all subsets of the ground set with at most K elements. Optimization with a uniform matroid constraint is equivalent to imposing cardinality constraints on the solution size, which is the standard TSO problem.

4.2.2 Linear matroid

Given a function $\phi : \mathscr{X} \to \{0,1\}^M$, the independent sets of a linear matroid induced by ϕ are all subsets $X \subseteq \mathscr{X}$ such that the vectors $\{\phi(x)\}_{x \in X}$ are linearly independent.

Application to coverage: Consider a setting where we require each robot to focus on a different region. Define the regions as M node subsets $S_m \subseteq \mathscr{V}$, and define the 'focus' of a path as the index of the region which contains the most nodes of the path (with ties broken deterministically). Let m_ρ be the smallest index corresponding to a subset which path ρ focuses on, that is $|S_{m_\rho} \cap \rho| \geq |S_m \cap \rho|$ for $m = 1, \ldots, M$. Now define

$\phi(\rho)$ as the m_ρth canonical basis vector in $\mathbb{R}^M$. Requiring the solution to be an independent set of the binary matroid induced by ϕ and with ground set $\mathscr{X}(p_s,\omega)$, enforces the desired diversity of focus.

4.2.3 Transversal matroid

Given subsets $\{\mathscr{X}_m\}_{m=1}^M$ of the ground set, the independent sets of the transversal matroid are all subsets X which are partial transversals of $\{\mathscr{X}_m\}_{m=1}^M$. In other words, if $X \in \mathscr{I}$, we can assign each element $x_i \in X$ a unique number $m_i \in \{1,\dots,M\}$ such that $x_i \in \mathscr{X}_{m_i}$.

Application to launch constraints: Consider a scenario where only one robot can start on each outgoing edge of v_s. This could arise for example when robots are aerial vehicles which must launch from runways, and only one can launch from a runway in each direction. Let $N(v_s)$ be the set of nodes with an edge to v_s, and define the subsets $\mathscr{X}_m = \{\rho \in \mathscr{X}(p_s,\omega) \mid \rho(1) = n_m\}$ for $n_m \in N(v_s)$. That is, we have one subset for each starting edge. Requiring that the solution is an independent set of the transversal matroid induced by $\{\mathscr{X}_m\}_{m=1}^M$ enforces the launch constraints.

Application to heterogeneous teams: Suppose we have M types of robots, a robot of type m has feasible path set $\mathscr{X}_m$, and that we can deploy at most K_m robots of type m. Requiring the solution to be an independent set of a transversal matroid induced by K_m copies of $\mathscr{X}_m$ for $m = 1,\dots,M$ enforces that no more than K_m robots of type m are selected.

Application to sub-graph diversity: Suppose we are given sub-graphs $\mathscr{G}_m = (\mathscr{V}_m, \mathscr{E}_m)$ for $m = 1,\dots,M$ with $v_s, v_t \in \mathscr{V}_m$, $\mathscr{V}_m \subseteq \mathscr{V}$, $\mathscr{E}_m \subseteq \mathscr{E}$, and we require that no more than K_m robots take paths in sub-graph $\mathscr{G}_m$ (we call this a 'sub-graph diversity' constraint). Let $\mathscr{X}_m$ correspond to the set of feasible paths in sub-graph $\mathscr{G}_m$. Requiring that the solution is an independent set of a transversal matroid induced by K_m copies of the sets $\mathscr{X}_m$ enforces the sub-graph diversity constraint.

Application to risk constraints: Suppose we have M survival probability thresholds $p_s^1 \le \cdots \le p_s^M$. This setting could arise when there is a constraint on the *risk* of many robot failures, but requiring uniform survival probability thresholds would be too conservative to visit all of the nodes. Then we can choose $\{p_s^m\}_{m=1}^M$ in order to provide the necessary flexibility while still maintaining tight control on risk. Requiring the solution to be an independent set of the transversal matroid induced by $\{\mathscr{X}(p_s^m,\omega)\}_{m=1}^M$ enforces the desired constraints.

4.2.4 Gammoid

A gammoid is induced by a directed graph $D(S,E)$ with a subset of nodes corresponding to elements in the ground set, e.g., $\mathscr{X} \subseteq S$, and a subset $U \subseteq S$. We say that two sets of nodes $X, Y \subseteq S$ are *linked* if $|X| = |Y|$ and there are $|X|$ node-disjoint paths from X to Y. The independent sets of a gammoid induced by D, U are all subsets $X \subseteq \mathscr{X}$ such that some subset of U is linked to X.

Application to nested cardinality constraints: Consider a simple setting where the ground set is partitioned by the sets $\{\mathscr{X}_1, \mathscr{X}_2\}$ and we may choose up to 2 items from $\mathscr{X}_1$, 2 items from $\mathscr{X}_2$ and 3 items total. The independent sets of a gammoid induced by the multi-partite graph in Figure 4.2 satisfy these constraints.

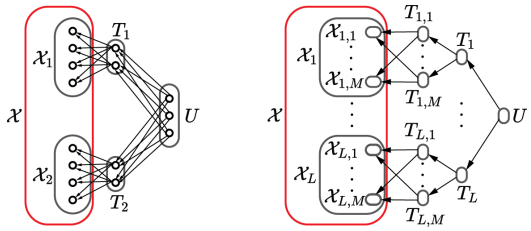


Figure 4.2: Illustration of multi-partite graphs which form gammoids. Left: An illustration of the graph with two layers of cardinality constraints. Right: An illustration of the graph with three layers. Boxes represent clusters of nodes, and lines represent edges which connect each node of the right cluster to each node of the left cluster.

This setting is easily extended to more complicated scenarios. For the MTSO, we could first partition based on robot types, then by sub-graphs, and finally by risk thresholds (or in a different order). An illustration of the graph for three layers of partitioning (e.g., robot types, sub-graphs, and team size) is shown in Figure 4.2. It is not necessary for the node groups to form a partition.

Note that this form of a gammoid is also known as a *laminar matroid*.

4.2.5 Truncation

Given $K \in \mathbb{N}$ and a matroid $(\mathscr{X}, \mathscr{I})$, let $\mathscr{I}' := \{I \in \mathscr{I} \mid |I| \leq K\}$ which is the set of independent sets with at most K elements. Then $(\mathscr{X}, \mathscr{I}')$ is a matroid and is called the K-truncation of $(\mathscr{X}, \mathscr{I})$. If the maximum team size is K, we can represent this constraint in addition to any of the scenarios above by using the K-truncation of the appropriate matroid.

4.3 The Greedy Sub-Problem

In this section we discuss the greedy sub-problem which is a building block for our solution algorithms. We give a description of our solution approach in Section 4.3.1, discuss performance considerations for its applications in Section 4.3.2.

4.3.1 Objective and algorithm

Given a (possibly empty) previously selected set of paths, $X_{L-1} = \{\rho_\ell\}_{\ell=1}^{L-1}$, the greedy sub-problem requires us to find a path from the set $\mathscr{X}_F(X_{L-1}, \mathscr{I})$ which maximizes the sum of path-dependent node rewards $v(\rho, j) \geq 0$ (corresponding to the discrete derivative of the appropriate objective function). Solving the greedy sub-problem is a critical step in both of our solution algorithms for the MTSO, but it is a very difficult task as it requires maximizing *path dependent* node rewards subject to a budget constraint.Specifically the path dependence is introduced by the risky traversal model, since the reward gained for visiting node j depends on

the probability that the robot successfully visits node j, which in turn depends on the path taken to the node. Furthermore, we must ensure that the path chosen is in $\mathscr{X}_F(X_{L-1}, \mathscr{I})$.

We address the path dependence by using a path *independent* approximation, $\hat{v}(j)$, of $v(\rho, j)$ which satisfies, for some $\gamma \in (0, 1]$:

$$v(\rho, j) \leq \hat{v}(j)\mathbb{I}_j(\rho) \leq \frac{1}{\gamma} v(\rho, j).$$

We address the matroid constraint by noting that the feasible set, $\mathscr{X}_F(X_{L-1}, \mathscr{I})$, can always be partitioned into sets $\{\mathscr{X}_m\}_{m=1}^M$, where $\mathscr{X}_m$ is the subset of paths in $\mathscr{X}$ which have all nodes and edges in a corresponding sub-graph $\mathscr{G}_m$. This is apparent from the fact that for each $\rho \in \mathscr{X}_F(X_{L-1}, \mathscr{I})$, we can construct a sub-graph $\mathscr{G}_\rho$ which has exactly the nodes and edges in ρ. Since a path is defined as a *unique* list of nodes and edges, and since feasible paths must start at v_s and end at v_t, the sub-graph $\mathscr{G}_\rho$ contains only one feasible path, ρ. In practice we can often partition $\mathscr{X}_F(X_{L-1}, \mathscr{I})$ using a very small number of sub-graphs, as detailed in Section 4.3.2.

Algorithm 3 Pseudocode for solving the sub-problem

procedure SOLVESUBPROBLEM(X_{L-1},$\hat{v}$)
  for $\mathscr{G}_m$ in `Partition`(X_{L-1}) **do**
    Form $\mathscr{G}_{m,O}$ from $\mathscr{G}_m$
    $\hat{\rho}_m \leftarrow$ `Orienteering`($\mathscr{G}_{m,O}, \hat{v}$)
  end for
  return $\underset{\rho \in \hat{\rho}_1, \ldots, \hat{\rho}_M}{\arg\max} \sum_{j=1}^V \hat{v}(j)\mathbb{I}_j(\rho)$
end procedure

Our approach to solving the sub-problem is given as pseudocode in Algorithm 3. We begin by calling the `Partition` routine, which returns a list of sub-graphs $\{\mathscr{G}_m\}_{m=1}^M$ which partitions the feasible set $\mathscr{X}_F(X_{L-1}, \mathscr{I})$. For each sub-graph, we form a log-transformed graph $\mathscr{G}_{m,O}$ which has edge weights $\omega_o = -\log(\omega)$. This graph and the approximate node reward $\hat{v}$ are passed to an `Orienteering` oracle, which finds a path which has at least fraction $1/\lambda$ the optimal reward with edge costs at most $B = -\log(p_s)$ (we discuss this oracle in depth in Section 2.2.2). This ensures that the path is of high quality and satisfies the survival probability threshold constraint. Finally the routine returns the best of the paths computed (with respect to either $\hat{v}$ or v, depending how hard v is to compute). The following lemma guarantees that the path returned has similar value as the optimum:

Lemma 10 (Sub-problem guarantee)**.** *Consider an MTSO problem with node weights d_j, let ρ be a feasible path in $\mathscr{X}_F(X_{L-1}, \mathscr{I})$, and $\hat{\rho}_L$ be the path output by the `SolveSubproblem` routine described in Section 4.3.1. Then*

$$\sum_{j=1}^V d_j v(\hat{\rho}_L, j) \geq \frac{\gamma}{\lambda} \sum_{j=1}^V d_j v(\rho, j).$$

Proof. By definition of the partition $\{\mathscr{X}_m\}_{m=1}^M$, for any $\rho \in \mathscr{X}_F(X_{L-1}, \mathscr{I})$ there is a set indexed by m_ρ such

that $\rho \in \mathscr{X}_{m_\rho}$. We have from the definitions of $\hat{\rho}_L$, $\hat{v}(\rho, j)$, and the `Orienteering` routine:

$$\sum_{j=1}^{V} d_j v(\rho, j) \leq \sum_{j=1}^{V} d_j \mathbb{I}_j(\rho)\hat{v}(j) \leq \lambda \sum_{j=1}^{V} d_j \mathbb{I}_j(\hat{\rho}_L)\hat{v}(j)$$
$$\leq \lambda \left(\frac{1}{\gamma} \sum_{j=1}^{V} d_j v(\hat{\rho}_L, j) \right).$$

□ □

Suppose the complexity of the `Orienteering` routine given any sub-graph is bounded by C_O, the complexity of the `Partition` routine is bounded by C_P, and there are at most M sub-graphs. Then the complexity of the `SolveSubproblem` routine is $C_P + MC_O$. Note that, as discussed in Section 2.2.3, the orienteering problem has many viable solution approaches despite being NP-hard. We describe the complexity of the partition routines below.

4.3.2 Efficiently partitioning the feasible set

For many examples we can partition the feasible set using a small number of sub-graphs, as detailed below.

Coverage – We can partition the sets for the coverage example by adding a constraint to a mixed integer formulation which requires the solution to have a specific focus. In this case M would be the number of regions to be covered.

Launch constraints – Given a set X_k, all paths which take edges $(v_s, \rho_\ell(1))$, $\ell = 1, \ldots, k$ are infeasible. Hence $M = 1$ and the sub-graph is the graph induced by the edges $\mathscr{E}' = \mathscr{E} \setminus \{(v_s, \rho_\ell(1))\}_{\ell=1}^{k}$ and takes at most $O(|\mathscr{E}|)$ operations to form.

Heterogeneous teams – The feasible sets are already partitioned by graphs $\mathscr{G}_m$ corresponding to the graph that a robot of type m uses. Given a set X_k, the partition routine returns all sub-graphs $\mathscr{G}_m$ for which X_k has fewer than K_m paths.

Sub-graph diversity – If the feasible path sets $\mathscr{X}_m$ are disjoint, then the routine simply returns any sub-graph for which X_k has fewer than K_m paths in $\mathscr{X}_m$. If the feasible path sets are not disjoint, then the routine solves an assignment problem to see whether an additional robot can be assigned a path in $\mathscr{X}_m$ without violating the constraints. The routine then returns any of the corresponding sub-graphs $\mathscr{G}_m$ which robots can still be assigned a path from. The complexity of each assignment is $O(K^{2.5})$ (using the Hopcraft-Korp algorithm), and there are at most MK assignments, giving complexity $O(MK^{3.5})$.

Risk constraints – Given a set of paths X_k, the partition routine finds the smallest value $\hat{m}$ such that we can add a path with survival probability threshold $p_s^{\hat{m}}$ (this can be done naively with complexity $O(M^2)$). Then the routine returns the sub-graph induced by the node set $V_{\hat{m}} := \{j \in \{1, \ldots, V\} \mid \zeta_j \geq p_s^{\hat{m}}\}$, which are all nodes reachable within the desired survival probability threshold. Note that in this case we also must change the budget used by the `Orienteering` routine to $-\log(p_s^{\hat{m}})$.

Nested cardinality constraints – Given a set of paths X_k, the partition routine tests whether a path can be added to each of the sets in the deepest layer (which partitions $\mathscr{X}$). If the sets are disjoint, this can be done in linear time, otherwise an assignment routine is run as in the sub-graph diversity example. Then a sub-graph corresponding to each of the subsets which we can still assign a path from is returned. The manner in which these sub-graphs are computed depends on the application, as described in each of the sections above.

4.3.3 Lazy implementation

Each time a sub-problem is solved, we can use Theorem 2 to give a bound on the value of any path in the corresponding sub-graph. Since the value of a path is a decreasing function as the algorithm progresses (due to submodularity), these bounds can be used in the following iterations. The *lazy* variant of the greedy algorithm solves the most promising sub-problem to get an initial solution, and then only solves sub-problems which have bounds that exceed the value of this initial solution. In our experiments this allowed us to skip solving 75% of the sub-problems.

4.4 The Approximate Greedy Algorithm

In this section we describe a greedy algorithm for solving the MTSO problem. We describe the objective function and detailed algorithm in Section 4.4.1, then give performance guarantees in Section 4.4.2, and characterize the complexity in Section 4.4.3.

4.4.1 Objective and algorithm

The greedy algorithm operates by iteratively selecting a path which maximizes the discrete derivative of the objective function for the MTSO given the set of previously chosen paths $X_{L-1} \subset \mathscr{X}$. A node-wise decomposition of this objective was given in Chapter 3 as,

$$\Delta J(\rho \mid X_{L-1}) = \sum_{j=1}^{V} d_j \mathbb{E}[z_j^L(\rho)] p_j(0, X_{L-1}),$$

which can be interpreted as the weighted sum over the probabilities that robot L is the first to visit node j. In the context of Lemma 10 we have $\nu(\rho, j) = \mathbb{E}[z_j^L(\rho)] p_j(0, X_{L-1})$, which can be approximated by setting $\hat{\nu}(j) = p_j(0, X_{L-1})$, with $\gamma = p_s$. Pseudocode for the approximate greedy algorithm is given in Algorithm 4. The algorithm alternates between updating the approximate node rewards, $\hat{\nu}$, and calling the `SolveSubproblem` routine to find an approximately optimal solution to the greedy sub-problem.

4.4.2 Guarantees

In this section we combine the results from Section 2.1.3 and 4.3.1 to prove that the value of the output of the `MGreedySurvivors` algorithm is close to the value of the optimal solution to the MTSO problem.

Algorithm 4 Approximate greedy algorithm for solving the MTSO problem.

procedure MGREEDYSURVIVORS($\mathscr{G}, \mathscr{M}$)
  for $j = 1, \ldots, V$ **do**
    $\hat{v}_1(j) \leftarrow d_j$
  end for
  $\rho_1 \leftarrow$ SolveSubproblem$(\emptyset, \hat{v}_1)$
  for $k = 1, \ldots, K-1$ **do**
    $\mathbb{E}[a_0^k(\rho_k)] \leftarrow 1$
    for $n = 1, \ldots, |\rho_k|$ **do**
      $\mathbb{E}[a_n^k(\rho_k)] \leftarrow \mathbb{E}[a_{n-1}^k(\rho_k)]\omega(e_{\rho_k}^n)$
      $\hat{v}_{k+1}(\rho_k(n)) \leftarrow (1 - \mathbb{E}[a_n^k(\rho_k)])\nu_k(\rho_k(n))$
    end for
    $\rho_{k+1} \leftarrow$ SolveSubproblem$(\{\rho_\ell\}_{\ell=1}^k, \hat{v}_{k+1})$
  end for
end procedure

Theorem 6 (Approximate greedy guarantee)**.** *Let X^* be an optimal solution to the MTSO problem, and $\hat{X}_K$ the set output by the* MGreedySurvivors *routine. If the survival probability constraint is p_s and each orienteering sub-problem is solved within constant factor $1/\lambda$, then the weighted expected number of nodes visited by a team of robots following the paths $\hat{X}_K$ is at least a fraction $\frac{p_s}{p_s+\lambda}$ of the optimum, that is:*

$$\sum_{j=1}^{V} \mathbb{E}\left[H_j(V_j(\hat{X}_K))\right] \geq \frac{p_s}{p_s+\lambda} \sum_{j=1}^{V} \mathbb{E}\left[H_j(V_j(X^*))\right].$$

Proof. Setting $\nu(\rho, j) = \mathbb{E}[z_j^L(\rho)]p_j(0, X_{L-1})$ and $\hat{v}(j) = p_j(0, X_{L-1})$, we have from Lemma 10 that the SolveSubproblem routine solves the sub-problems within factor $\alpha = p_s/\lambda$. Now applying Lemma 4 and Theorem 2 we have the desired result. □

In many scenarios robots are either valuable or difficult to replace, and so operators will choose survival probability thresholds p_s which are close to 1. Combined with an effective oracle routine, Theorem 6 guarantees that the greedy approach never collects less than approximately half of the optimal reward. This is a strong statement, since solving this problem optimally is extraordinarily difficult (especially as the team size grows).

4.4.3 Complexity

The MGreedySurvivors routine has complexity $O(KV^2 + K(MC_O + C_P))$, where the first term is the complexity of updating the V weights K times (each update costs as most V flops), and the second term is the complexity of calling the SolveSubproblem routine K times. In typical scenarios, the complexity will be dominated by KMC_O, as solving the orienteering problem is typically several orders of magnitude more difficult than partitioning the feasible set or updating the weights. Each of the M orienteering problems can be solved independently, so by leveraging parallel computation the complexity will scale as $O(KC_O)$.

4.5 The Accelerated Continuous Greedy Algorithm

In this section we describe an accelerated continuous greedy algorithm for solving the MTSO problem. We begin by describing the objective function and detailed algorithm in Section 4.5.1, then give performance guarantees in Section 4.5.2 and characterize the complexity in Section 4.5.3.

4.5.1 Objective and algorithm

The accelerated continuous greedy algorithm [17] performs a discretized coordinate gradient ascent which optimizes the multilinear extension along 'coordinates' corresponding to independent sets in the matroid. The state of the algorithm is given by a sparse vector $y \in [0,1]^{|\mathscr{X}|}$ which can be interpreted as a probability distribution over elements in $\mathscr{X}$. The algorithm runs for $1/\delta \in \mathbb{Z}_+$ steps (indexed by i), each consisting of K iterations (indexed by ℓ). During every iteration a single component of y corresponding to a path $\rho \in \mathscr{X}$ is incremented by δ, that is $y \leftarrow y + \delta \mathbf{1}_\rho$ where $\mathbf{1}_\rho$ is the indicator vector which has all zero entries except the component corresponding to ρ, which is 1. The component ρ is selected to ensure that (1) the paths corresponding to the set of components updated in a given step are independent, and (2) the component approximately maximizes $F(y+\delta\mathbf{1}_\rho) - F(y)$, which is equivalent to maximizing the discretized gradient with discretization step δ.

The objective for the ACGA can be written as an expectation with respect to the random set $R(y)$, which contains elements in $\mathscr{X}$ sampled independently with probability y_ρ, that is:

$$\mathbb{P}\{R(y) = X\} = \prod_{x \in X} y_x \prod_{x' \in X^c} (1 - y_{x'}).$$

The following lemma gives an alternate expression for the objective in terms of a sum over path dependent node rewards:

Lemma 11 (Objective function for the ACGA)**.** *Let f be the objective of the MTSO and F its multilinear extension. The objective for the ACGA with state y during the ℓth iteration of the ith step can be written as*

$$F(y+\delta\mathbf{1}_\rho) - F(y) = \delta \sum_{j=1}^{V} \frac{d_j \mathbb{E}[z_j^\ell(\rho)]}{1 - y_\rho \mathbb{E}[z_j^\ell(\rho)]} \mathbb{E}[p_j(0, R(y))].$$

This result is derived directly from the definitions of the relevant variables (a detailed proof is given in the appendix).

Pseudocode for our algorithm is given in Algorithm 5, which consists of three main parts: (1) efficiently computing $\mathbb{E}[p_j(0,R(y))]$, (2) removing path dependence for the sub-problem, and (3) rounding the solution. We discuss each part in detail below.

1) Efficiently computing $\mathbb{E}[p_j(0,R(y))]$: In general, computing an expectation over functions of the random set $R(y)$ is difficult because the function must be evaluated for every realization of $R(y)$, and for our setting there are $2^{K/\delta}$ possible realizations of $R(y)$. In our case, we can exploit the product form of $p_j(0,R(y))$

to iteratively compute the expected value for fixed sizes of $R(y)$, that is $\mathbb{E}[p_j(0,R(y))\mathbb{I}\{|R(y)| = m\}]$, which are then summed to compute the desired expectation. The algorithm is given in pseudocode in Algorithm 6 and details on its correctness are given in the appendix. The complexity of this approach is $O(VK^2\delta^{-2})$.

2) Removing path dependence for the sub-problem: The objective is path dependent because the term $\frac{\mathbb{E}[z_j(\rho)]}{1-y_\rho\mathbb{E}[z_j(\rho)]}$ depends on the probability that node j is visited (which is a function of the path taken) and the weight assigned to path ρ. We handle this term in two stages: we start by assuming that $y_\rho = 0$ (which is true for most paths) in which case we have $\frac{\mathbb{E}[z_j(\rho)]}{1-y_\rho\mathbb{E}[z_j(\rho)]} = \mathbb{E}[z_j(\rho)]$ and use the strategies developed in Section 4.4.1 to approximately maximize the objective by solving an orienteering problem. We then compute the value of the $O(K/\delta)$ paths which have $y_\rho > 0$ explicitly and select the better of these paths or the output of the orienteering routine solved in the first stage. We show in the Appendix that this two-stage approach produces a path with value within factor p_s/λ of the optimal.

3) Rounding the solution: We use the `SwapRounding` procedure to round y in order to get our final solution. In order to ensure that we achieve a good result with probability 1, we repeat the rounding procedure until the result is at least $(1-\delta)F(y)$. The expected number of calls to the rounding routine is upper bounded by $1/(1-e^{-p_s\delta^2/8})$, which is $O(\delta^{-2})$ (we defer details of these calculations to the appendix).

Algorithm 5 Approximate continuous greedy algorithm for solving the MTSO problem.

procedure ACGA($\mathscr{G}, \mathscr{M}, \delta$)
  $y \leftarrow \vec{0}$
  for $i = 1, \ldots, 1/\delta$ **do**
    $X(i) \leftarrow \emptyset$
    for $\ell = 1, \ldots, K$ **do**
      $\hat{v} \leftarrow$ `UpdateWeights`(y)
      $\rho' \leftarrow$ `SolveSubproblem`$(X(i), \hat{v})$
      $\rho \leftarrow \underset{\rho', \{\rho \in \mathscr{X}_F(X(i),\mathscr{I}) : y_\rho > 0\}}{\arg\max} \sum_{j=1}^{V} d_j v(\rho, j)$
      $y_\rho \leftarrow y_\rho + \delta$
      $X(i) \leftarrow X(i) \cup \rho$
    end for
  end for
  while True **do**
    $\hat{X} \leftarrow$ `SwapRounding`(y)
    if $f(\hat{X}) \geq (1-\delta)F(y)$ **then return** $\hat{X}$
    end if
  end while
end procedure

4.5.2 Guarantees

We give a guarantee for the algorithm described in Section 4.5.1 by using a similar approach as [17]:

Theorem 7. *Let X^* be an optimal solution to the MTSO problem and $\hat{X}$ be the output of the ACGA routine*

Algorithm 6 Efficient weight update routine.

```
1:  procedure UpdateWeights(y)
2:      for j = 1, ..., V do
3:          w = 0⃗_{|y|+1}
4:          w_0 ← 1
5:          for x : y_x > 0 do
6:              for m = |y|, ..., 1 do
7:                  w_m ← w_{m-1} · y_x · (1 − E[z_j(x)]) + w_m · (1 − y_x)
8:              end for
9:              w_0 ← w_0 · (1 − y_x)
10:         end for
11:         v̂_j ← Σ_{m=0}^{|y|} w_m
12:     end for
13: end procedure
```

with parameters δ and λ. Then the value of the set $\hat{X}$ is lower bounded by a constant factor of the optimum:

$$f(\hat{X}) \geq f(X^*)(1-\delta)\left(1-\exp\left(-\frac{p_s}{\lambda+\delta p_s}\right)\right).$$

The detailed proof is given in the appendix. The basic idea is to first use the properties of the `SolveSubproblem` routine to lower bound the incremental increase in the multilinear extension between subsequent steps of the algorithm, then to use a recursive argument to show that $F(y)$ satisfies the desired inequality, which in turn ensures that $f(\hat{X})$ satisfies the guarantee.

Tightness of the guarantee – As the approximation parameters converge to their ideals ($\delta \to 0, \lambda \to 1, p_s \to 1$), the guarantee converges to $f(\hat{X}) \geq f(X^*)(1-e^{-1})$, which matches the hardness bounds for the general problem of maximizing a submodular function subject to a matroid constraint. We also note that for $\delta \ll \lambda$ the guarantee is approximately $1-e^{-p_s/\lambda}$, which matches the guarantee given for the TSO problem.

4.5.3 Complexity

The ACGA routine calls the `UpdateWeights` routine K/δ times, solves K/δ sub-problems, and calls the `SwapRounding` routine an average of $O(\delta^{-2})$ times. Hence the total complexity is

$$O(\delta^{-1}K(MC_O + C_P) + \delta^{-3}K^3V + \delta^{-3}K),$$

and since generally $C_O \gg \delta^{-2}K^2V$, this is $O(\delta^{-1}VMC_O)$, which is a factor of δ^{-1} greater than the approximate greedy routine.

4.6 Experimental Results

In this section we validate and compare our solution approaches using simulations. In particular, we demonstrate the rich sets of constraints a matroid can model in Section 4.6.1, where we consider a challenging environmental monitoring scenario and compare two choices for the `Orienteering` oracle. Section 4.6.2 quantifies the empirical scaling of our algorithm using synthetic data. In particular, section 4.6.3 shows that the `MGreedySurvivors` algorithm scales as expected as the team size and number of sub-problems required to partition the feasible set grows; and Section 4.6.4 shows that the ACGA algorithm scales as expected as δ shrinks. Finally in Section 4.6.5 we compare the `MGreedySurvivors` and ACGA approaches for a range of δ values and discuss the strengths of each approach.

4.6.1 Environmental monitoring application

We consider the application illustrated in Figure 4.1: the Coral Triangle has a diverse ecosystem and the goal is to use a robotic fleet with multiple sensor configurations to monitor the wildlife populations in 'Marine Protected Areas'. We use piracy incident data [2] and model attacks as a Poisson random variable in a manner similar to [41]. We selected 106 marine protected areas from [1] as nodes in a complete graph and computed the shortest (minimum risk) paths between each pair of sites to find the edge weights. In our scenario, we can deploy up to 25 robots of three types (at most twelve of each type), and each robot type has a different utility in each region. Each region can support the traffic of at most three robots of each type.

We require that the expected losses over the worst 5% of outcomes be no more than five failed robots (this is called the Conditional Value at Risk, and denoted $\text{CVaR}_{0.05}$). For the data shown, the most difficult node to reach requires survival probability 0.64, but if we use a uniform survival probability threshold $p_s = 0.64$, the risk is unacceptably high ($\text{CVaR}_{0.05} = 16.26$). We satisfy the constraint $\text{CVaR}_{0.05} \leq 5$ by setting $p_s^1, \ldots, p_s^5 = 0.6$, and $p_s^6, \ldots, p_s^{25} = 0.859$, that is we allow five robots to take more risky paths and constrain the rest to more conservative paths. All of the constraints above can be represented using a single gammoid, and the feasible set can be partitioned using at most $M = 15$ sub-graphs. Note how expressive gammoids are - this example uses just three levels of 'nested constraints' (robot types, traffic limits, and risk constraints) but we could easily use more.

We consider two choices for the `Orienteering` routine: an open-source Variable Neighborhood Search (VNS) heuristic produced by HeuristicLab [31] and a mixed-integer program (MIP) formulation solved using CPLEX with a 5% tolerance. We observe that the VNS heuristic gives near-optimal paths, and solves all of the sub-problems for each of the twenty five robots in 51 seconds on a quad-core 4GHz processor with 32GB RAM available. The MIP oracle takes 60 seconds and yields nearly the same results. While the MIP oracle can find high quality paths quickly, as the tolerance is decreased the computation time dramatically increases. Optimally solving the sub-problems does not necessarily lead to better overall behavior, as shown in Figure 4.3. The upper bound shown is computed as the smaller of 1) the upper bound found using the greedily selected paths and the approximation ratio, and 2) the sum of all node rewards $\sum_{j=1}^{V} d_j = 106$. This upper

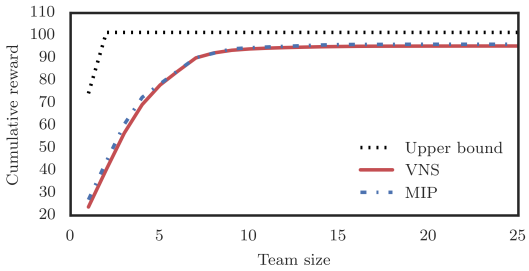


Figure 4.3: Cumulative reward for paths planned by `MGreedySurvivors` using the MIP and VNS `Orienteering` as oracle routines. Both approaches compute high quality sets of paths, though VNS is somewhat faster.

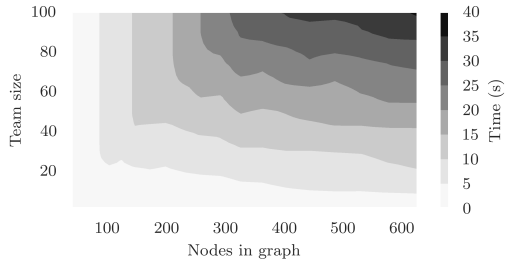


Figure 4.4: Scaling of `MGreedySurvivors` as K and V grow. Data shown is the median of 110 samples, and agrees with an $\Theta(MKC_O)$ trend.

bound is loose because it ignores the constraints (particularly on the traffic flow through a region) which becomes apparent as the team size grows larger. Note that although the theoretical approximation guarantee is 0.36, a team of 10 robots achieves 0.928 the maximum possible reward, despite only planning one path at a time.

4.6.2 Synthetic problem generation

We demonstrate the scaling properties of the `MGreedySurvivors` and ACGA algorithms using synthetic data. Nodes are placed randomly on a 2D plane and clustered into groups of 15-20 nodes (this ensures the complexity of solving the orienteering problem, C_O, is constant). We consider a sub-graph diversity constraint with sub-graphs induced by the clusters. Edge weights are chosen so that $-\log(\omega(e))$ is proportional to the length of the edge e, the survival probability threshold is set to 0.7, and we use a MILP formulation for the planar orienteering problem with tolerance 5%.

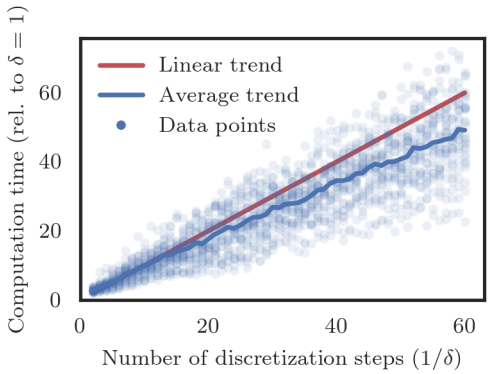


Figure 4.5: Scaling of the run time of the ACGA routine relative to the `MGreedySurvivors` routine as the number of discretization steps increases. Note that run time increases approximately linearly with δ^{-1}, as predicted in Section 4.5.3.

4.6.3 Sub-problem complexity scaling

We demonstrate the complexity of the `MGreedySurvivors` algorithm with respect to K and M using the synthetic dataset described above. We use the lazy variant of the greedy algorithm [18], which only solves a sub-problem if it is not dominated by another (this allows us to skip 76% of the sub-problems in our experiment). The median computation times are shown in Figure 4.4. The trends agree with the $O(MKC_O)$ scaling predicted in Section 4.4.3, and our approach solves very large problems with very large teams in under a minute.

4.6.4 Discretization complexity scaling

As discussed in Section 4.5.3, we expect the run time of the ACGA to be roughly δ^{-1} times that of the `MGreedySurvivors` routine (which is ACGA with $\delta = 1$). We demonstrate this trend empirically by using the synthetic data generated as described at the beginning of this section (with $V = 82$ and $K = 8$). For each problem we compute the baseline by averaging the run time of 5 calls to the `MGreedySurvivors` routine. We then measure the run times of the ACGA with the desired δ values and use the ratio of ACGA run times to the baseline run time as our data points. We repeated this for 30 randomly generated problems and show the results in Figure 4.5. The randomness in run times is due to several factors (1) lazy implementation allows for problems to be skipped when good bounds are available, (2) MIP solver time can depend significantly on the particular problem, and (3) MIP solver times are not deterministic. Note that the average run time (with respect to the random problem instances) actually grows *sub-linearly*, likely due to the lazy implementation of the ACGA.

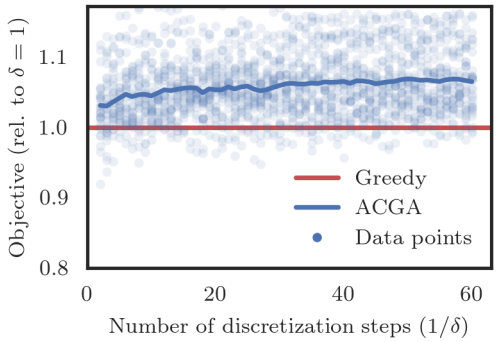


Figure 4.6: Comparison of objective achieved by the ACGA and `MGreedySurvivors` routines with $\delta^{-1} \in \{2,\ldots,60\}$ for 30 random MTSO problems. Note that as δ^{-1} increases the ACGA more consistently outperforms the baseline and average performance increases.

4.6.5 Effect of discretization on performance

Using the same data as in Section 4.6.4, we analyze the influence of the discretization parameter on the performance of ACGA relative to `MGreedySurvivors` in terms of the value of their solutions. As shown in Figure 4.6, increasing δ^{-1} generally improves the performance, and generally increases the chance that the result produced by ACGA will be better than the result produced by `MGreedySurvivors`. Note that the benefits of smaller discretization sizes drop off faster than the increase in computation time, so in practice a relatively small value of δ^{-1} (e.g., 8 or 16) should be used. In scenarios where computation time is paramount, the `MGreedySurvivors` routine provides a fast way of achieving high quality solutions.

4.7 Extensions

In this section we discuss additional extensions and applications of our work, presented in order of the maturity of available results. In Section 4.7.1, we describe p-systems and their applications to robotics. Minor modifications to algorithms presented above have constant factor guarantees for p-systems. In Section 4.7.2 we describe the coverage variant of the MTSO, which is closely related to the results given in Chapter 6. Results from [25] can be applied to provide guarantees for our ACGA algorithm in this setting.

4.7.1 p-system constraints

Consider the MTSO where the matroid $\mathscr{M}$ is replaced by a p-system. Recall that a p-system is an independence system which is downward closed and every base (maximal independent set) has at most p elements. The crucial distinction between p-systems and matroids is that the latter has the exchange property, which implies that *every* base in a matroid has the same number of elements (which we called the rank of the matroid).

Applications to robotics

The notion of p-systems allows us to model many common constraints in robotics.

Collision avoidance – Consider a setting where we do not allow two robots to simultaneously traverse the same edge in the graph (this also requires us to introduce a notion of time, which we discuss in Chapter 3). One can easily verify that collision avoidance is downward closed - any subset of a set of collision free paths will also be collision free. The value of p will be the smaller of the out-degree of the starting vertex and the number of robots allowed.

Capacity constraints – Collision avoidance can be generalized to a capacity constraint, where each edge has a capacity, and each robot has a demand. We can require that the sum of demands for robots simultaneously traversing an edge must be below the capacity of the edge. Note that collision avoidance is a special case where all demands and capacities are unit. This scenario can model road networks as well as communications networks, where a video sensor may demand many more resources than other types of sensors.

Matroid intersections – The intersection of p matroids can also be represented as a p-system. This can allow us to simultaneously enforce risk constraints, sensor availability, and sub-graph diversity constraints. Note that this is different from the nested cardinality constraints used in the experiments. For example, we could require that at most 5 of the 25 paths planned are risky and that at most 12 of each robot type can be selected (rather than the nested case, which would say, e.g., at most 12 of the risky paths can be of a given type, and at most 12 of the safe paths can be of a given type).

Algorithms and guarantees

We can use a slightly modified version of our algorithms above to find a set which is in the independent family of sets for a p-system and (approximately) maximizes the expected number of nodes visited. The only difference is that now the feasible set $\mathscr{X}_F(X,\mathscr{I})$ is defined with respect to the independent family of sets for a p-system. A similar application-specific partitioning technique can be used in the `SolveSubproblem` routine.

The guarantee we state in Theorem 1 is a special case of the more general statement from Appendix B of [24], which for a p-system gives the constant factor guarantee $\frac{\alpha}{\alpha+p}$. For our applications, we will typically constrain teams to have at most K robots, meaning that the guarantees become $\frac{p_s}{p_s+K\lambda}$, which while a constant factor, quickly becomes loose as the team size grows.

4.7.2 Multiple objectives and coverage problems

Consider the problem where we are given a set of submodular functions $f_1,\ldots,f_N$, corresponding values $V_1,\ldots,V_N$, and want to find a set X such that $f_n(X)\geq V_n$ for $n=1,\ldots,N$.

Applications to robotics

This problem could model a coverage variant of the MTSO, where we are seeking a set X which satisfies a matroid constraint while also ensuring that the probability node j is visited satisfies a given threshold, $p_v(j)$. In this setting we define the functions $f_j(X) = \min\{1 - p_j(0,X), p_v(j)\}$ and set $V_j = p_v(j)$. This is similar to the coverage variant of the TSO given in [61], where we seek the smallest team which satisfies the desired visit probabilities.

Algorithm and guarantees

The solution algorithm is a modified version of the continuous greedy algorithm and is outlined in [25]. Theorem 2.5 from [25] guarantees that the output of the modified algorithm satisfies $F_n(y) \geq (1 - e^{-1})V_n$ for $n = 1, \ldots, N$, or returns a certificate of infeasibility. In our case the ACGA has a weaker $1 - e^{-p_s/\lambda}$ guarantee, and so the result would be modified accordingly. If the visit probability thresholds are small enough, we can guarantee feasibility by dividing V_j by $(1 - e^{-p_s/\lambda})$, ensuring that the visit probability is greater than $p_v(j)$.

4.8 Conclusions

We formulate the *Matroid Team Surviving Orienteers* (MTSO) problem, where we seek a set of paths which forms an independent set of a matroid, maximizes the expected number of nodes visited by at least one robot, and ensures the probabilities each robot reaches its destination are above a threshold. This problem is a significant generalization of the Team Surviving Orienteers problem discussed in Chapter 3, and is distinct from previous work because it combines a submodular objective, chance constraints, and matroid constraints. We give numerous applications of matroids to robotic path planning such as coverage, launch constraints, limits on the number of available robots of multiple types, restrictions on the amount of traffic which can flow through a region, and combinations of the above. The MTSO is particularly challenging to solve because of the risky traversal model (where a robot might not complete its planned path) which creates a complex, history-dependent coupling between the edges chosen and the distribution of nodes visited. We present two solution algorithms: an approximate greedy algorithm for solving the MTSO problem which guarantees that its output achieves at least $1 - e^{-p_s/\lambda}$ of the optimal reward, and a variant of the ACGA which guarantees that its output achieves at least $\simeq 1 - e^{-p_s/\lambda}$ the optimal reward. The algorithms rely on a partitioning routine to satisfy the matroid constraints, and we show numerous examples where our algorithm runs in polynomial time. We demonstrate the efficiency of our approaches by applying it to a scenario where a team of robots must gather information while avoiding pirates in the Coral Triangle.

Chapter 5

The Risk Sensitive Coverage Problem

In this chapter we consider the dual to the TSO problem, where instead of maximizing the number of nodes visited by a given team, we seek the smallest set of routes (or set with fewest expected failures) which satisfies a set of visit probability constraints. For example, consider a search and rescue mission where a team of robots searches for victims trapped in a storm, and the probability that a robot survives while traversing between two sites (corresponding to likely locations of victims) depends on the weather. The team operator seeks to find a set of paths for the smallest team which guarantees that sites are searched with given probability thresholds, and that the probabilities each robot returns safely satisfy a survival threshold.

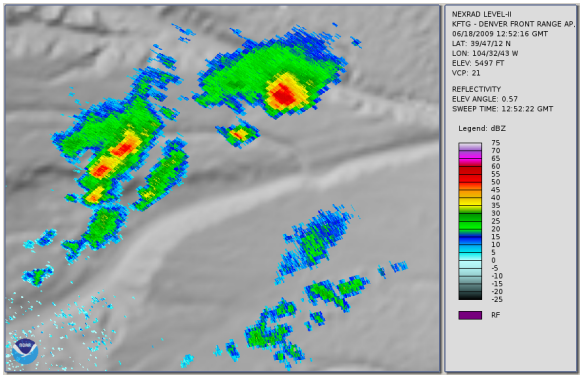


Figure 5.1: Illustration of the "base reflectivity" of a storm, which can be used to infer the probability robots survive traversing between sites. Data from NOAA NEXRAD level II dataset, and visualization courtesy the Weather and Climate Toolkit [3].

The challenge facing the operator can be posed as a *set cover* problem, where it must choose the smallest set of paths which satisfy a coverage constraint (e.g. the search/visit probability thresholds). We call our formulation the Risk-Sensitive Coverage (RSC) problem, and while the set cover problem is well understood [26, 22, 62], the RSC problem is more challenging because 1) the ground set of paths is exponentially large in

the number of nodes and edges, and 2) the notion of *risky traversal* (where a robot may fail while traversing an edge) precludes many existing approaches used for path planning because it introduces a complex interaction between the edges chosen and the probability a node is visited.

The key insight in this chapter is that the visit probability constraints can be represented using a submodular function which allows us to precisely state the problem as a *submodular set cover* problem, and justifies using a cost-benefit greedy algorithm to find a solution. We develop an algorithm for the special case of unit costs and a more general cost-benefit path planning algorithm with applications beyond the RSC. We provide approximation guarantees and complexity bounds for both algorithms. Although existing work considers the set cover problem with exponentially large ground sets [63] or submodular routing problems [39, 7] separately, our work is novel because we combine the two and give a computationally efficient algorithm with approximation guarantees.

A handful of variants of the vehicle routing problem consider stochastic coverage constraints similar to the RSC. A setting where travel time between nodes is stochastic and nodes must be visited within certain time windows is considered by [64]. Their approach seeks to ensure that the probability that each node is visited within its time window is above a given threshold, but their algorithm does not explicitly enforce this constraint and in practice many of their simulations do not achieve the desired guarantees. While this setting could be used to model the RSC by setting the deadlines to be the negative logarithm of the survival probability threshold and treating the negative logarithm of the survival probabilities as "time", our approach explicitly enforces the coverage constraint and provides approximation guarantees. Another vehicle routing problem with deadlines and travel time uncertainty is considered by [65], where the objective is to minimize the sum of the constraint violations. While this is nominally related, our problem is quite different because we enforce the visit probability constraints for *each* node and minimize the team size, rather than fixing the team size and minimizing *total* visit probability constraint violation.

Coverage problems in robotics [66] seek a set of routes which guarantee some notion of coverage. Coverage problems are in general NP-hard, since they generalize the orienteering problem [4], which seeks a maximum weight path in a graph that satisfies a budget constraint. The persistent monitoring problem [67] seeks a set of cyclic routes which minimize the time between visits to a node. The authors provide a linear program which can be solved efficiently and give theoretical guarantees concerning the robustness of their solution. Another broad category of coverage problems is the informative path planning problem (IPP), where the goal is to plan paths which gather as much information as possible about an environmental random variable. The IPP is related to our setting because information is also a submodular function, but typically IPP problems are stated as maximization problems rather than coverage problems. The literature on coverage problems is vast, however to our knowledge the risky traversal model, which is integral to this work, is not used by existing work. We note that although [68] proposes a similarly named problem, the notion of "risk" in their work is probability of detection, which does not interact with the coverage constraints in the same manner as in our setting.

Given a ground set of items, a cost function, and a submodular coverage function, the *submodular set*

cover problem seeks a minimum cost subset which saturates the coverage function. The problem was posed and solved using a cost-benefit greedy algorithm by [26] which iteratively builds a solution by adding the item with the best ratio of benefit to cost. Several $1 + \log(\Delta)$ approximation guarantees (where Δ is a problem-dependent parameter) were given, which guarantee that the cost of the optimum and approximate solutions are nearly the same. Matching hardness bounds were given by [22] and [26]. A more in-depth treatment of the submodular set cover and submodular knapsack problem is given by [62], which considers enhanced algorithms which exploit the curvature of the submodular coverage function in order to provide refined analysis. The approaches taken by [26, 62] assume that the ground set can be efficiently searched, but for our setting this is not the case. Exponentially large ground sets were considered by [63], who provide strong guarantees for the special case when the submodular function is integral and the ground set is the family of independent sets of a matroid (which extends linear independence to set functions). Their work does not generalize, because it relies on specific results for optimizing over independent sets of matroids and integral submodular set functions. Large ground sets were also considered by [7] from a submodular function maximization perspective, where the goal is to select edges which form a path (rather than to select paths which saturate a submodular function).

Our approach to solving the RSC relies on solving a cost-benefit path planning problem, where we (approximately) maximize the ratio of benefits (new coverage) to costs (e.g. failure probability). A closely related multi-objective problem is the *Travelling salesman problem with profits* (TSPP) which asks for a route which maximizes the rewards of nodes visited while simultaneously minimizing the distance travelled. Since it is a bi-objective problem, its solution is the Pareto optimal set (i.e., solutions which are not 'dominated' by another). There are two prevailing methods to construct the Pareto optimal set for bi-objective routing problems. The ε-constraint method repeatedly optimizes one objective while constraining the value of the second objective. This approach was applied to the TSPP by [69, 70].

The two-phase approach constructs the Pareto set by first finding all 'efficient solutions' (lying on the convex hull of the Pareto optimal set) and then finding the non-efficient solutions. The second phase is typically much harder, but uses the "efficient solutions" to reduce the search space. Of particular note are the BE algorithm [71] and the ABE algorithm [70], both of which solve sub-problems in order to partition the 'objective space' (space of possible objective values) into smaller subregions which are easier to search. Algorithmically this is very similar to our approach, although the TSPP assumes integral profits and a solution is the entire (possibly exponentially large) Pareto optimal set. In contrast, we allow for scalar profits and are interested in a particular (non-linear) combination of 'profit' (benefit) and 'distance' (cost).

Statement of Contributions. The contribution of this chapter is sixfold. First, we propose the Risk-Sensitive Coverage (RSC) problem and show that it is an instance of the submodular set cover problem. By considering risky traversal, we extend the state of the art for robot coverage problems, and by considering an exponentially large ground set we extend the state of the art in the submodular set cover problem. Second, we provide a linear relaxation which allows us to implement a cost-benefit greedy algorithm efficiently by utilizing standard orienteering problem solvers. Third, we provide a bi-criteria approximation guarantee, which

ensures that the cost of the set output by our routine is close to the optimum for a closely related problem. Specifically, our result states that the solution with L robots is no more than $\frac{\lambda}{p_s}(1+\log(\lambda\Delta_L/p_s))$ times the optimum solution cost for a problem satisfying at least L/K fraction of the constraints, where K is the size of the final output of our algorithm, p_s is the per-robot survival probability threshold, $1/\lambda \leq 1$ is the approximation guarantee for an oracle routine which solves the orienteering problem, and Δ_L is the ratio of incremental coverage gain from the first and Lth path planned. Fourth, we provide an optimization routine for solving a broad class of cost-benefit path planning problems. Specifically, we consider the problem max $f_B(\rho) - f_C(\rho)$, where $f_B(\rho)$ typically represents the benefit and $f_C(\rho)$ typically represents the cost of path ρ. Our approach requires that optimizing $f_B(\rho)$ and $f_C(\rho)$ separately is equivalent to optimizing the sum of carefully selected edge or node weights. This assumption is fairly mild, and allows us to consider challenging non-linear combinations of cost and benefit such as $(\sum b_j)/(1-\prod c_i)$, $(\prod b_j) + C\prod c_i$, and $(\sum b_j) + \sqrt{\sum c_i^2}$, where c_i are cost coefficients and b_j are reward coefficients, with i and j indexing edges or nodes in the path (depending on the application). Fifth, we bound the number of calls to an orienteering problem oracle required to reach a desired error. Specifically, after $O(1/\varepsilon)$ calls to the oracle (with oracle tolerance δ), the absolute error is at most $\varepsilon + \delta$ and the relative error (assuming the value of the initial solution, $\hat{F}_0$, is positive) is at most $1 + \varepsilon + \delta/\hat{F}_0$. If the cost and benefits are known to reside in a finite interval, then these results imply that the corresponding cost-benefit path planning problem is of similar complexity as solving an orienteering problem. Finally, we demonstrate the quality of the paths selected by our routine. For a special case of the RSC (for which we could compute optimal solutions) our solution uses at most 33% more robots than the optimum. When tested on edge-sparse graphs, our cost-benefit planning approach scales empirically with the square of the number of nodes in the graph and provides near-optimal solutions (exhaustive search scales exponentially in the number of nodes). We then apply our routine to a search and rescue scenario with 225 nodes and visit probability thresholds of 0.95. Our routine finds a set of 36 paths which satisfy the constraints with approximation ratio 9. The first 13 of these paths satisfy 80.7% of the constraints with approximation ratio 4.33. This chapter is an extension of the conference paper [61].

5.1 Problem Statement

We present the formal statement of the RSC problem in Section 5.1.1, its equivalence to submodular set cover in Section 5.1.2, and the cost-benefit path planning problem in 5.1.3. We then discuss an example of the RSC in Section 5.1.4 and its applications in Section 5.1.5.

5.1.1 The RSC problem

We use the same notation and setting as set out in Chapter 2. Let $c : \mathscr{X} \to \mathbb{R}_+$ be a cost function, $h_j : \mathbb{Z}_+ \to \mathbb{R}_+$ a reward function (as defined in Section 3.1.1). We define the expected reward accumulated at node j as

$$\bar{H}_j(X) := \sum_{m=0}^{|X|} h_j(m) p_j(m, X).$$

We assume that h_j satisfies the properties of Lemma 3.1.4, which implies $\bar{H}_j(X)$ is a submodular function.

Let H_j, $j = 1, \ldots, V$ be the minimum reward to be collected at node j. The Risk-Sensitive Coverage problem is formally stated as:

> **Risk-Sensitive Coverage problem:** Given a graph $\mathscr{G}$, edge weights ω, survival probability threshold p_s, cost function c, reward functions $\{h_j\}_{j=1}^V$, and reward thresholds $\{H_j\}_{j=1}^V$, minimize the team cost while satisfying all constraints:

$$\begin{aligned}
\underset{K, \rho_1, \ldots, \rho_K}{\text{minimize}} \quad & \sum_{k=1}^{K} c(\rho_k) & \\
\text{subject to} \quad & \mathbb{P}\left\{ a^k_{|\rho_k|}(\rho_k) = 1 \right\} \geq p_s & k = 1, \ldots, K \\
& \rho_k(0) = v_s & k = 1, \ldots, K \\
& \rho_k(|\rho_k|) = v_t & k = 1, \ldots, K \\
& \bar{H}_j(\{\rho_k\}_{k=1}^K) \geq H_j & j = 1, \ldots, V
\end{aligned}$$

The objective is to minimize the cost of the routes chosen. The first set of constraints enforces the survival probability, the second and third sets of constraints enforce the initial and final node constraints. The last constraint is the coverage constraint and requires that the reward collected at node j is at least H_j. The RSC should be considered as a higher level model for routing a team of robots subject to visit constraints. In practice lower level routines will be necessary to handle issues such as collision avoidance and robot dynamics.

We are primarily interested in two classes of cost functions. When costs are uniform (i.e. $c(\cdot) = 1$) then the optimal solutions will use as few robots as possible. When costs are the failure probability (i.e. $c(\rho) = 1 - \mathbb{E}[z_{v_t}(\rho)]$) then an optimal solution will minimize the expected number of failed robots. In principle our approach can be applied to many other settings.

5.1.2 Equivalence to submodular set cover

In this section we show that the RSC is equivalent to the submodular set cover problem with ground set $\mathscr{X}$, which contains finitely many copies of each path in $\mathscr{X}(p_s, \omega)$. A subset $X \subset \mathscr{X}$ automatically satisfies the first three sets of constraints for the RSC (survival probability, start and end nodes). Next we give a

submodular function $g : 2^{\mathscr{X}} \to \mathbb{R}_+$ such that $g(X) = g(\mathscr{X})$ if and only if the reward thresholds are satisfied.

Define the functions f_j for $j = 1, \ldots, V$ to be the minimum of the expected reward collected at node j by robots following the paths in X and the reward threshold:

$$f_j(X) := \min\left\{\bar{H}_j(X), H_j\right\}.$$

We assume that h_j satisfies the conditions of Lemma 3.1.4 hence $\bar{H}_j$ is a submodular function, and by composition rules f_j is also submodular. Now we define the coverage function,

$$g(X) = \sum_{j=1}^{V} f_j(X),$$

which is also a submodular function of the set of paths. If the constraints are satisfied by $X \subset \mathscr{X}$, then $f_j(X) = H_j = f_j(\mathscr{X})$ which implies $g(\mathscr{X}) = g(X)$. If $g(X) = g(\mathscr{X})$ and there is some subset $Y \subseteq \mathscr{X}$ which satisfies the constraints, then $g(X) = \sum_{j=1}^{V} H_j$, which implies that the set X satisfies the coverage constraints. Hence $g(X) = g(\mathscr{X})$ is equivalent to saying that the set X satisfies the coverage constraints, so the RSC problem is equivalent to

$$\begin{aligned} \underset{X \subseteq \mathscr{X}}{\text{minimize}} \quad & \sum_{\rho \in X} c(\rho) \\ \text{subject to} \quad & g(X) = g(\mathscr{X}) \end{aligned}$$

which is a submodular set cover problem with domain $\mathscr{X}$, cost function c, and coverage function g. Motivated by the strong performance of the cost-benefit greedy algorithm for the submodular set cover problem [26], we proceed to study the cost-benefit path planning problem in detail.

5.1.3 Cost-benefit path planning

Let $c_e \in \mathbb{R}$ be the *cost* of taking edge $e \in \mathscr{E}$, and $b_j \in \mathbb{R}$ be the *benefit* of visiting node $j \in \mathscr{V}$. Define $f_C : \mathscr{X} \to \mathbb{R}_{\geq 0}$ as the *path* cost function, and $f_B : \mathscr{X} \to \mathbb{R}$ as the *path* benefit function. We are interested in cost and benefit functions which satisfy,

$$\underset{\rho \in \mathscr{X}}{\text{Maximize}}\ f_B(\rho) \iff \underset{\rho \in \mathscr{X}}{\text{Maximize}} \sum_{j \in \rho} b_j.$$

$$\underset{\rho \in \mathscr{X}}{\text{Minimize}}\ f_C(\rho) \iff \underset{\rho \in \mathscr{X}}{\text{Minimize}} \sum_{e \in \rho} c_e,$$

Namely, maximizing $f_B(\rho)$ is equivalent to maximizing the sum of the benefits for nodes in ρ and minimizing $f_C(\rho)$ is equivalent to minimizing the sum of edge costs in ρ. We refer to this as 'additive equivalence'. The cost-benefit path planning problem is then:

Cost-benefit Path Planning Problem: Given a set of paths $\mathscr{X}$, functions f_B, f_C with corresponding coefficients $\{b_j\}_{j\in\mathscr{V}}$, $\{c_e\}_{e\in\mathscr{E}}$, and an error bound ε, solve

$$\underset{\rho\in\mathscr{X}}{\text{Maximize}}\ f(\rho) := f_B(\rho) - f_C(\rho)$$

within error bound ε.

The error bound constraint can be either absolute, meaning $f(\rho) \geq f(\rho^*) - \varepsilon$, or relative, meaning $f(\rho^*) \leq (1+\varepsilon)f(\rho)$. By imposing the additive equivalence, we retain enough structure to meaningfully solve the problem, and by allowing the cost and benefit functions to have different forms we can handle a broad class of challenging routing problems.

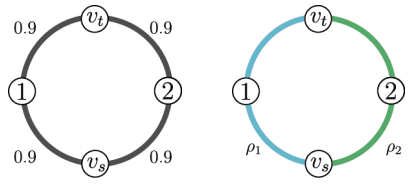
Remark: Our approach and formulation of the cost-benefit path planning problem can be generalized to handle node costs, edge rewards, negative rewards (i.e. bi-cost problems) or negative costs (i.e. bi-objective problems). To keep the presentation simple we mainly focus on the cost-benefit trade-off which has positive edge costs and positive node rewards, as this is the case that is useful for the RSC.

5.1.4 Example

An example of the RSC problem with single visit reward functions and uniform costs is given in Figure 5.2(a). There are four nodes and four edges. The survival threshold is $p_s = 0.8$ and the edge weights are all 0.9. There are two feasible paths, $\rho_1 = \{v_s, 1, v_t\}$, $\rho_2 = \{v_s, 2, v_t\}$, and the probability a robot reaches node v_t is 0.81 for either path. For visit probability thresholds $p_v(1) = p_v(2) = 0.9$, it is easy to verify that the optimal solution is $\{\rho_1, \rho_2\}$. For visit probability thresholds $p_v(1) = p_v(2) = 0.99$ the optimal solution is $\{\rho_1, \rho_1, \rho_2, \rho_2\}$.

5.1.5 Applications and variants of the RSC

Heterogeneous teams The RSC is easily extended to a scenario where there are M different types of robots available, each with different cost functions $c(m, \rho)$ and capabilities represented by feasible sets $\mathscr{X}_m$. The



(a) Graph $\mathscr{G}$ (b) Feasible Paths

Figure 5.2: Example of the Risk-Sensitive Coverage problem. (a) The graph has four nodes and four edges. The probability of surviving a given edge is 0.9. (b) For survival probability threshold 0.8, there are two feasible paths from node v_s to v_t: $\rho_1 = \{v_s, 1, v_t\}$ and $\rho_2 = \{1, 2, v_t\}$.

objective function is now the sum of the costs of the types of robots selected, and the constraints are now that each path be in the respective feasible set of the robot type. This can model a wide variety of scenarios, for example when some lower cost robots can be exposed to more risk than more expensive variants; or when each robot type can only visit a subset of the nodes in the graph, e.g. in underwater scenarios where nodes may correspond to different depths.

Edge variants We can extend to an *edge* coverage scenario by re-defining all node variables as edge variables as described in Section 3.2.4. This is useful for applications such as scientific missions where edges correspond to routes which the robots gather information along, and the objective is to find the smallest team which can complete the mission with the desired coverage.

Arbitrary terminal nodes We can consider the problem where the robots can end at any subset of nodes by placing an edge with weight 1 between these nodes and node v_t, and removing all other edges which end at v_t. This would model a scenario where there are multiple depots where the robots may end at which are equally desirable.

Fuel limits We can finally consider a variant where we have a second edge weight ω_2 which represents fuel consumption, and paths must not exceed a fuel budget B_F. This introduces a new constraint that the sum of edge weights along a path is less that B_F.

5.2 Solution Approach

In this section we study how to solve the RSC problem. In Section 5.2.1 we derive a node additive decomposition of the constraint function and show that we can approximately maximize the coverage of a path by solving an orienteering problem. We then discuss the cost-benefit sub-problem in detail in Section 5.2.2, with an analysis of its convergence and complexity. We give the top-level algorithm in Section 5.2.3, and provide an analysis of its complexity. We give a bi-criteria guarantee on the quality of our solution in Section 5.2.4 and discuss modifications for the extensions in 5.2.5.

5.2.1 Additive equivalent objective

Given a previously selected set of paths $\hat{X}_{L-1}$, the cost-benefit greedy sub-problem for the RSC problem requires us to find a path $\hat{\rho}_L$ which maximizes the ratio of discrete derivative of the coverage function at $\hat{X}_{L-1}$ with respect to $\hat{\rho}_L$ to cost. This section lays the groundwork for how to optimize the cost-benefit ratio (which we discuss more in Section 5.2.2). We show that the coverage function can be written as a sum of (path dependent) node rewards, and can be effectively approximated with path *independent* weights.

By construction, the set $\mathscr{X}$ has (finitely) many copies of each path in $\mathscr{X}(p_s, \omega)$, so $\mathscr{X} \setminus \hat{X}_{L-1}$ always contains at least one copy of each feasible path. Since the discrete derivative of g is identical for each copy

we can reduce the search domain to paths in $\mathscr{X}(p_s, \omega)$. We use a similar approach as in Chapter 3 to linearize the constraint function so it can be represented as a sum of path independent node weights.

The discrete derivative of g is the sum of the discrete derivatives of f_j. If the reward threshold has been satisfied, then $\Delta f_j(\rho \mid \hat{X}_{L-1}) = 0$, otherwise it is the smaller of $H_j - \bar{H}_j(X)$ and $\Delta\bar{H}_j(\rho \mid X)$. For $\phi \in [0,1]$, define

$$\delta_j(\phi, X) = \begin{cases} 0, \text{ if } \bar{H}_j(X) \geq H_j \\ \min\left\{ H_j - \bar{H}_j(X), \phi \sum_{m=1}^{|X|} \Delta h_j(m) p_j(m-1, X) \right\}, \text{ else .} \end{cases}$$

It is easy to verify that $\Delta f_j(\rho \mid X) = \delta_j(\mathbb{E}[z_j(\rho)], X)$, and if $p < p'$ then $\delta(p, X) \leq \delta(p', X)$. Let $\mathbb{I}_j(\rho)$ be equal to 1 if node j is in ρ and 0 otherwise. From the definition of δ_j and feasibility of ρ, we have the following inequalities:

$$p_s \mathbb{I}_j(\rho)\delta_j(\zeta_j, X) \leq \Delta f_j(\rho \mid X) \leq \mathbb{I}_j(\rho)\delta_j(\zeta_j, X).$$

This means that the path *independent* quantity $\delta_j(\zeta_j, X)$ is a good characterization of the path *dependent* quantity $\Delta f_j(\rho \mid X)$, especially when p_s is close to unity. We form our approximate greedy algorithm by using this path independent approximation, where we are looking to maximize the sum:

$$\Delta\bar{g}(\rho \mid \hat{X}_{L-1}) := \sum_{j=1}^{V} \mathbb{I}_j(\rho)\delta_j\left(\zeta_j, \hat{X}_{L-1}\right),$$

which represents an *optimistic* estimate of the discrete derivative of the coverage function at $\hat{X}_{L-1}$ with respect to ρ.

We can find the path which (approximately) maximizes reward (but not the ratio) by solving an orienteering problem on the graph $\mathscr{G}_O$, which has the same edges and nodes as $\mathscr{G}$ but has edge weights $\omega_O(e)$ and node rewards $v_L(j) = \delta_j(\zeta_j, \hat{X}_{L-1})$. Solving the orienteering problem on $\mathscr{G}_O$ with budget $-\log(p_s)$ will return a path that maximizes the sum of node rewards (which is $\Delta\bar{g}(\rho \mid X_{L-1})$), and satisfies $\sum_{e\in\rho} -\log(\omega(e)) \leq -\log(p_s)$, which is equivalent to $\mathbb{P}\{a^L_{|\rho|}(\rho) = 1\} \geq p_s$. The following lemma characterizes the value of such a path:

Lemma 12 (Single robot constant-factor guarantee)**.** *Suppose we are given a set of paths $X \subset \mathscr{X}$. Let* `Orienteering` *be a routine that solves the orienteering problem within constant-factor $1/\lambda$, that is for node weights $v(j) = \delta_j(\zeta_j, X)$, path $\hat{\rho}$ output by the routine, and any path $\rho \in \mathscr{X}(p_s(r), \omega_r)$,*

$$\sum_{j=1}^{V} \mathbb{I}_j(\hat{\rho})v(j) \geq \frac{1}{\lambda} \sum_{j=1}^{V} \mathbb{I}_j(\rho)v(j).$$

Then for any $\rho \in \mathscr{X}(p_s(r), \omega_r)$ we have

$$\Delta g(\hat{\rho} \mid X) \geq \frac{p_s}{\lambda} \Delta g(\rho \mid X)$$

Proof. We have by the properties of δ_j and the `Orienteering` routine,

$$\Delta g(\rho \mid X) \leq \sum_{j=1}^{V} \mathbb{I}_j(\rho)\delta_j(\zeta_j, X) \leq \lambda \sum_{j=1}^{V} \mathbb{I}_j(\hat{\rho})\delta_j(\zeta_j, X)$$
$$\leq \frac{\lambda}{p_s} \sum_{j=1}^{V} \Delta f_j(\hat{\rho} \mid X) \leq \frac{\lambda}{p_s} \Delta g(\hat{\rho} \mid X)$$

□

Intuitively, this lemma follows because if p_s is close to unity, the probability that a robot *could* visit a node is not too far from the probability that a robot *actually* visits the node. Note that this is very similar to the guarantee in Lemma 5, however the new proof is necessary because the latter does not consider truncated rewards (i.e. the $\min\{\cdot\}$ operator).

5.2.2 Cost-benefit sub-problem

When costs are uniform, the cost-benefit greedy sub-problem reduces to maximizing $\Delta g(\rho \mid \hat{X}_L)$. Hence using Lemma 12 we have that solving an orienteering problem with node weights $\delta_j(\zeta_j, \hat{X}_L)$ is a $\frac{p_s}{\lambda}$ approximate algorithm for the cost-benefit greedy sub-problem.

When costs are not uniform, we solve the cost-benefit path planning problem by searching over the two-dimensional objective space. Level curves in this space are lines with unit slope, since they imply $f_B(\rho) = f_C(\rho) + d$ for some constant d. Structurally our search is very similar to the ε-constraint and two phase approaches commonly used to construct the Pareto optimal set. Our algorithm can be interpreted as an implicit search over the Pareto set of cost-benefit optimal paths, where it focuses on solutions which may optimize the specific scalarization $f_B(\rho) - f_C(\rho)$. Our algorithm relies on two sub-routines:

$$\begin{aligned}
&\text{SR1:} \quad \underset{\rho \in \mathscr{X}}{\text{Minimize}} \quad f_C(\rho), \\
&\text{SR2:} \quad \underset{\rho \in \mathscr{X}}{\text{Find}} \quad \rho \\
&\qquad \text{Such that } f_C(\rho) \leq C, \\
&\qquad\qquad f_B(\rho) \geq \max_{\rho^* : f_C(\rho^*) \leq C} f_B(\rho^*) - \delta.
\end{aligned}$$

SR1 is used to bound the cost of any optimal solution to the cost-benefit path planning problem and SR2 is used to bound the benefit of any optimal solution (by setting $C = \infty$) and to improve the bound on benefit for paths with cost less than C. Because the functions f_C and f_B have additive equivalents, these subroutines are equivalent to well-known graph optimization problems. In the case with positive edge costs and positive node benefits, SR1 is equivalent to a standard shortest path problem and SR2 is equivalent to the orienteering problem. Pseudocode for the algorithm is given in Algorithm 7, with an illustration of the search procedure in Figure 5.3.

Algorithm 7 Algorithm for solving the general cost-benefit path planning problem.

procedure CBPP($\mathscr{G}$, f_B, f_C,$\{b_j\}$,$\{c_e\}$, ε)
  Compute additive equivalents $\{b_j\}$, $\{c_e\}$
  Compute ρ_C^*, ρ_B^*
  $\hat{\rho} \leftarrow \underset{\rho \in \{\rho_C^*, \rho_B^*\}}{\text{argmax}} \; f(\rho)$, $\hat{F} \leftarrow f(\hat{\rho})$
  $\bar{F} \leftarrow f_B(\rho_B^*) + \delta - f_C(\rho_C^*)$
  $R_0 \leftarrow \texttt{Region}(f_C(\rho_B^*), f_C(\rho_B^*), \bar{F})$
  Q.enqueue$(R_0, \bar{F}_{R_0})$
  while !Q.empty **do**
    $R \leftarrow Q$.pop
    if $\bar{F}_R - f(\hat{\rho}) < \varepsilon$ **then return** $\hat{\rho}$
    end if
    $\rho_R \leftarrow \texttt{Solve\_OP}(\mathscr{G}, R)$
    $\hat{\rho} \leftarrow \underset{\rho \in \{\rho_R, \hat{\rho}\}}{\text{argmax}} \; f(\rho)$, $\hat{F} \leftarrow f(\hat{\rho})$
    $R_L, R_U \leftarrow \texttt{Split}(R, f_B(\rho_R))$
    Q.enqueue$(R_L, \bar{F}_{R_L})$
    Q.enqueue$(R_U, \bar{F}_{R_U})$
  end while
  return Fail
end procedure

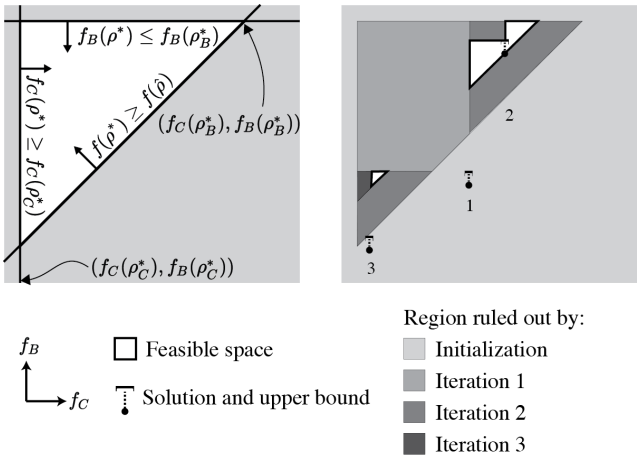


Figure 5.3: Illustration of the feasible space after initialization (left) and after three iterations of the refinement phase (right).

Initialization The algorithm begins by solving subroutines 1 and 2 to bound the feasible space (line 3). We denote a solution to subroutine 1 by ρ_C^*, and a solution to subroutine 2 (with $C = \infty$) by ρ_B^*. The current best solution $\hat{\rho}$ is set to the better of the two initial solutions (line 4) and the upper bound is then set to $\bar{F} = f_B(\rho_B^*) + \delta - f_C(\rho_C^*)$ (line 5).

From these initial solutions we have three inequalities which define a triangular region of feasible space, as illustrated in Figure 5.3. Namely, we have that any optimal solution ρ^* must satisfy

$$f_C(\rho^*) \geq f_C(\rho_C^*)$$
$$f_B(\rho^*) \leq f_B(\rho_B^*) + \delta$$
$$f(\rho^*) \geq f(\hat{\rho})$$

Structure of the feasible space As the algorithm progresses, the feasible space is fragmented into regions defined by disjoint intervals of cost values, a regional upper bound on the objective, and the global lower bound $f(\hat{\rho})$ (an example after three iterations is given in Figure 5.3). Note that each region is a right trapezoid with unit slope.

The additive error within region R is the difference between the regional upper bound and the global lower bound and denoted by $E_R := \bar{F}_R - \hat{F}$. Note that E_R is the length of the "tall" vertical side of the region (see Figure 5.4).

The tolerance of region R is the height of the "short" vertical side of a region, and denoted by δ_R. The tolerance is introduced by allowing additive error δ in solutions to SR2. We store the regions that compose the feasible space using a priority queue with the priority of R set to $\bar{F}_R$ (lines 7, 15, 16).

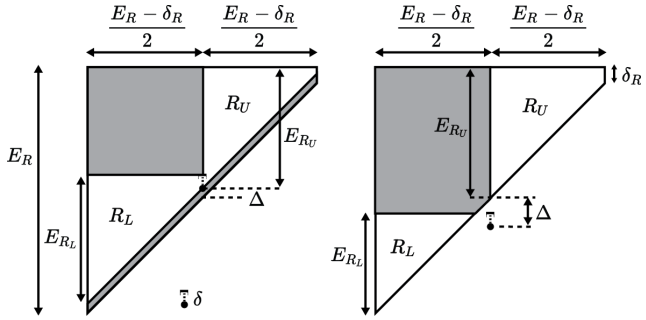


Figure 5.4: Illustration of how a solution to the orienteering problem (SR2) splits a feasible region for the cases when the solution produces a new lower bound (left) and does not (right).

Solution refinement The algorithm then iterates by selecting region R off the top of the queue (i.e., the region with the largest upper bound). If the error within this region is less than ε, then the routine returns $\hat{\rho}$ (line 10). Otherwise, it solves an orienteering problem with the cost bound set to the mid-point of the region's extent in cost-space (line 12). The resulting path splits R into two smaller regions, the lower half R_L

and the upper half R_U (illustrated in Figure 5.4). The `Split` routine takes in a region R and the solution to an orienteering problem ρ_R and returns R_L and R_U (line 14) with bounds set as shown in Figure 5.4.

Note that by construction the widths of R_L and R_U are half of the width of the parent region R, and the solution ρ_R either produces a new upper bound for region R_L or improves the global lower bound. Furthermore the tolerances δ_{R_L} and δ_{R_U} are bounded by $\max\{\delta, \delta_R\}$, and since the initial tolerance is δ, we have $\delta_R \leq \delta$ for any region.

Branch and bound algorithm analysis We begin by formalizing the relationship between E_{R_L}, E_{R_U} and E_R in terms of δ and the difference $\Delta := |f(\hat{\rho}) - f(\rho_R)|$ (evaluated before updating $\hat{\rho}$ on line 13).

Invariant 1: At any point in the algorithm, $E_R = E_{R_L} + E_{R_U} + \Delta - \delta$. Furthermore,

$$\max\{E_{R_L}, E_{R_U}\} \leq \frac{1}{2}(E_R + \delta).$$

The proof follows from geometry of triangles and is illustrated graphically in Figure 5.4. Note that this invariant implies that $E_{R_L} + E_{R_U}$ is largest (i.e. convergence is slowest) when $\Delta = 0$. Using this invariant we can provide a bound on the absolute error:

Lemma 13. *(Bound on absolute error) Let E_n denote the absolute error after solving $2^n - 1$ orienteering problems, with E_0 as the error after the initialization step. Then*

$$E_n \leq 2^{-n} E_0 + (1 - 2^{-n})\delta.$$

Proof. The slowest convergence is when $\Delta = 0$ for every sub-problem solved. In this case, the algorithm proceeds uniformly through the feasible regions and so the errors take one of two values at a given time. When $2^n - 1$ sub-problems have been solved, all regions have the same error. Our proof is by induction over n.

The base case follows immediately from invariant 1 since there is a single region $E_0 = E_R$ and so $E_1 = \max\{E_{R_L}, E_{R_U}\} \leq \frac{1}{2}(E_0 + \delta)$.

Step n : Since we are considering the worst case with $\Delta = 0$, E_{n-1} is the largest error after $2^{n-1} - 1$ calls to SR2. For each of the problems evaluated during step n we have $E_R \leq E_{n-1}$, and so applying Invariant 1 again we have

$$E_n \leq \frac{1}{2}(E_{n-1} + \delta) \leq 2^{-n} E_0 + (1 - 2^{-n})\delta.$$

□

An alternate way of characterizing this convergence is that it takes $O(1/\varepsilon)$ calls to the orienteering oracle to get absolute error bounded by $\varepsilon + \delta$.

We can use a similar approach to bound the relative error, which is the ratio of upper and lower bounds. We begin by characterizing how the relative error changes over a single iteration of the algorithm:

Invariant 2: If the initial lower bound $\hat{F}_0$ is positive, then

$$\max\left\{\frac{\bar{F}_{R_L}}{\hat{F}_{R_L}}, \frac{\bar{F}_{R_U}}{\hat{F}_{R_U}}\right\} \leq \frac{1}{2}\left(\frac{\bar{F}_R + \delta}{\hat{F}_R} + 1\right).$$

Proof. From Figure 5.4 and Invariant 1 we have

$$\bar{F}_{R_L} = \bar{F}_R - \left(\frac{E_R - \delta}{2}\right)$$
$$\bar{F}_{R_U} = \bar{F}_R - E_R + E_{R_L} \leq \bar{F}_R - \left(\frac{E_R - \delta_R}{2}\right)$$

both of which are bounded by $\frac{1}{2}(\bar{F}_R + \hat{F}_R + \delta)$. Since $\hat{F}_{R_L} = \hat{F}_{R_U} \geq \hat{F}_R$, taking the ratio of the bounds gives the desired result. □

Now we can provide a bound on the convergence rate of the relative error for the case when $\hat{F}_0 > 0$:

Lemma 14. *(Bound on relative error) Let $\bar{F}_n/\hat{F}_n$ be the relative error after solving $2^n - 1$ orienteering problems, and assume that the initial solution has positive value ($\hat{F}_0 > 0$). Then*

$$\frac{\bar{F}_n}{\hat{F}_n} \leq 2^{-n}\frac{\bar{F}_0}{\hat{F}_0} + (1 - 2^{-n})\left(1 + \frac{\delta}{\hat{F}_0}\right).$$

The proof is identical to the bound on the absolute error, except we use Invariant 2 in place of Invariant 1.

An alternate way of characterizing this convergence is that it takes $O(1/\varepsilon)$ calls to the orienteering solver to get a relative error bounded by $1 + \varepsilon + \frac{\delta}{\hat{F}_0}$. If $\hat{F}_0 \gg \delta$, this guarantee is quite strong.

5.2.3 Algorithm

We can now describe the complete routine for solving the RSC problem. Pseudocode for our algorithm is given in Algorithm 9. Our algorithm relies on two sub-routines. Define the method `Dijkstra`($\mathscr{G}$, i, j), which returns the length of the shortest path from i to j on the edge weighted graph $\mathscr{G}$ using Dijkstra's algorithm. Given an edge weighted graph $\mathscr{G}$, node rewards v, and cost function c, the `costbenefit`$(\mathscr{G}, v, c)$ routine solves the appropriate cost-benefit path planning problem within factor α. Pseudocode for the `costbenefit` routine is given in Algorithm 8. If costs are uniform, it solves a single orienteering problem, otherwise it solves an appropriate version of the cost-benefit path planning problem as described in Section 5.2.2. We begin by forming the graph $\mathscr{G}_O$ with log-transformed edge weights $\omega_O(e)$, and then use Dijkstra's algorithm

to compute the maximum probability that a node can be reached. The algorithm then proceeds to plan paths until the constraints are satisfied. During each iteration, we solve the cost-benefit path planning problem to greedily choose the path that maximizes $\Delta\bar{g}$. This path is added to the solution set, and the algorithm terminates if all constraints are satisfied.

Algorithm 8 Algorithm for solving the cost-benefit greedy sub-problem. If costs are uniform it solves a single orienteering problem, otherwise it performs an implicit search over the Pareto optimal set of paths (described in detail in Section 5.2.2).

procedure `COSTBENEFIT`($\mathscr{G}$, v, c)
  if Uniform costs **then**
    return `Orienteering`($\mathscr{G}$, v, $-\log(p_s)$)
  else
    return `CBPP`($\mathscr{G}$, $\log(\cdot)$, $\log(\cdot)$, v, ω, ε)
  end if
end procedure

Algorithm 9 Approximate greedy algorithm for solving the RSC problem.

procedure `CGREEDYSURVIVORS` ($\mathscr{G}, \{H_j\}, c$)
  Form $\mathscr{G}_O$ from $\mathscr{G}$, such that $v_s = 1$, $v_t = V$
  $\hat{X} \leftarrow \emptyset$
  for $j = 1, \ldots, V$ **do**
    $\zeta_j \leftarrow \exp(-$`Dijkstra`$(\mathscr{G}_O, 1, j))$
  end for
  while $\max_j \delta_j(\zeta_j, \hat{X}) > 0$ **do**
    $\hat{X} \leftarrow \hat{X} \cup$ `costbenefit`$(\mathscr{G}_O, \{\delta_j(\zeta_j, X)\}_{j=1}^{V}, c)$
  end while
  return $\hat{X}$
end procedure

Let $\bar{K}$ be an upper bound on the number of robots selected. This can be computed by using a naive policy which sends robots along the safest route to a node until the constraint is met, then repeats for all nodes. Suppose that the complexity of the `Orienteering` oracle is C_O. Then the complexity of our algorithm for uniform costs is $O(V^2 \log(V)) + O(V\bar{K}) + O(\bar{K}C_O)$. The first term is the complexity of calculating ζ_j for all nodes, the second term is the complexity of updating each weight $K \leq \bar{K}$ times, and the final term is the complexity of solving the $K \leq \bar{K}$ orienteering problems. For general costs with solver tolerance ε, the complexity is $O(\bar{K}C_O/\varepsilon)$.

5.2.4 Approximation guarantees

In this section we combine the results from Chapter 3, this chapter, and [26] to give approximation guarantees for the `CGreedySurvivors` algorithm. The set cover problem is not approximable within a constant factor [22] and our guarantees are solution dependent, meaning that the approximation ratio depends on the specific problem posed and solution found by our algorithm. We give a *bi-criteria* guarantee, which ensures that the

cost of our solution is no more than a factor $\gamma \geq 1$ greater than the minimum cost solution which satisfies fraction $\beta \leq 1$ of the constraints.

Theorem 8 (Bi-criteria approximation for the RSC)**.** *Given a RSC problem with coverage function g and costs c, let $\hat{X}_K$ be a solution found using an α–approximate cost-benefit greedy algorithm. For $L \leq K$, let $\hat{X}_L$ be the first L paths selected by the algorithm, and let X_L^* be a set with minimum cost which satisfies $g(X_L^*) \geq g(\hat{X}_L)$. Then the cost of the set $\hat{X}_L$ is bounded above by*

$$\sum_{\hat{\rho} \in \hat{X}_L} c(\hat{\rho}) \leq \frac{1}{\alpha}\left(1 + \log\left(\frac{\frac{1}{\alpha}g(\hat{X}_1)}{g(\hat{X}_L) - g(\hat{X}_{L-1})}\right)\right) \sum_{\rho^* \in X_L^*} c(\rho^*),$$

and the set $\hat{X}_L$ satisfies

$$g(\hat{X}_L) \geq \frac{L}{K} g(\mathscr{X}).$$

Proof. For a given solution $\hat{X}_L$, define the functions

$$\hat{f}_j(X) = \min\{f_j(\hat{X}_L), f_j(X)\}, \qquad j = 1, \ldots, V.$$

Note that $\hat{f}_j(X) \leq f_j(X)$, and $\hat{f}_j(\hat{X}_\ell) = f_j(\hat{X}_\ell)$ for $\ell \leq L$. By definition of $\hat{X}_L$, we have for $\ell \leq L$

$$\begin{aligned}\sum_{j=1}^{V} \Delta\hat{f}_j(\hat{x}_\ell \mid \hat{X}_{\ell-1}) &= \sum_{j=1}^{V} \Delta f_j(\hat{x}_\ell \mid \hat{X}_{\ell-1}) \\ &\geq \alpha \sum_{j=1}^{V} \Delta f_j(x \mid \hat{X}_{\ell-1}) \geq \alpha \sum_{j=1}^{V} \Delta\hat{f}_j(x \mid \hat{X}_{\ell-1}),\end{aligned}$$

which implies that the set $\hat{X}_L$ is an α–approximate cost-benefit greedy solution to the RSC with normalized, monotone, submodular coverage function $\hat{g}(X) := \sum_{j=1}^{V} \hat{f}_j(X)$. Hence we can apply Theorem 3 to get the first result. The second result follows since $g(\hat{X}_K) = g(\mathscr{X})$ and g is submodular. □

Using the results from Section 5.2.2 we can specify α for the two cases of cost-function we are primarily interested in:

Uniform cost case Applying the CBPP routine to the RSC problem with uniform costs we have from Lemma 12 that $\alpha = \frac{p_s}{\lambda}$.

Failure probability costs For the case where $c(\rho) = 1 - \mathbb{E}[z_{v_t}(\rho)]$, we have

$$f_C(\rho) = \log\left(1 - \prod_{e \in \rho} \omega_e\right),$$

which has additive equivalent weights $c_e = -\log(\omega_e)$. Combining Lemma 12 with Lemma 13 we have that after $O(1/\varepsilon)$ solutions to the orienteering problem,

$$\log\left(\frac{\Delta g(\hat{\rho})}{c(\hat{\rho})}\right) \geq \log\left(\frac{\Delta g(\rho^*)}{c(\rho^*)}\right) - \varepsilon - \log(\lambda/p_s),$$

which implies that $\alpha = e^{-\varepsilon}\frac{p_s}{\lambda}$.

Tightness of Guarantees The tightness of Theorem 8 for the case $L = K$ depends significantly on the given problem. Consider the example from Section 5.1.4, with visit probability $p_v(1) = p_v(2) = 0.9$. Then the greedy algorithm will choose $\hat{X}_1 = \{\rho_1\}$, $\hat{X}_2 = \{\rho_1, \rho_2\}$. For this simple graph $\alpha = 1$ (since we can try all feasible paths), and so the guarantee states that $|\hat{X}_2| \leq (1 + \log(0.9/0.9))|X_2^*| = |X_2^*|$, that is the upper bound implies optimality. However if we change the visit probability threshold for node 1 to $p_v(1) = 0.9 + \varepsilon$ for some small $\varepsilon > 0$, then the greedy algorithm terminates after choosing $\hat{X}_3 = \{\rho_1, \rho_2, \rho_1\}$. The guarantee at $L = K$ becomes $|\hat{X}_3| \leq (1 + \log(1 + 0.9/\varepsilon))|X_3^*|$, which can be arbitrarily loose as $\varepsilon \to 0$. However using the bi-criteria guarantee for $L = 2$, we have $|\hat{X}_2| \leq (1 + \log(0.9/0.9))|X_2^*| = |X^*|$, and furthermore $g(X_2^*) \geq g(\mathscr{X}) - \varepsilon$, meaning that the set $\hat{X}_2$ is an optimal solution to a problem which has nearly the same constraints.

5.2.5 Algorithm variants

Heterogeneous teams For the heterogeneous team setting, we solve one cost-benefit greedy sub-problem per robot type at each step of the greedy routine, and select the path/type which has the best value to cost ratio. The complexity increases linearly in the number of robot types.

Edge variants For the edge variant, we solve an *arc* orienteering problem [44] in place of the standard orienteering problem at each step of the cost-benefit greedy routine. The complexity remains the same, though the guarantees available for the arc orienteering problem are somewhat weaker than for the standard orienteering problem.

Arbitrary terminal nodes Arbitrary terminal nodes are represented using the graph, and so the underlying algorithm remains the same. Appropriate modifications for the graph were discussed in Section 5.1.5.

Fuel limits We can consider fuel costs by solving a *capacitated* orienteering problem as the greedy sub-problem (which have constant factor approximations, [45]), where capacities correspond to the fuel budget.

5.3 Simulations and Discussion

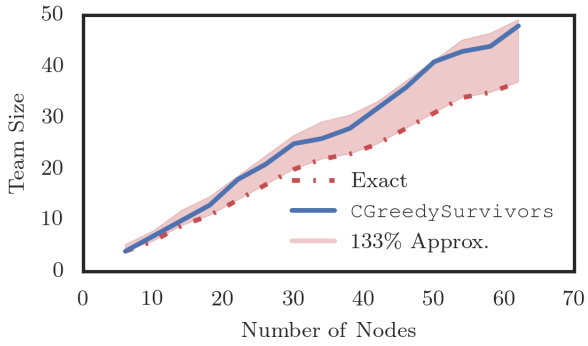


Figure 5.5: Performance comparison over different graph sizes for a special case of the RSC.

In this section we characterize the performance and efficiency of our approach. We consider complete graphs (with an edge between every pair of nodes) and visit probabilities 0.95, which represent challenging scenarios due to the number of edges in the graph and visits required.

5.3.1 Comparison to optimal policies

The RSC problem is a mixed integer nonlinear programming (MINLP) problem because of the nonlinear coverage constraint. However, we can formulate a special case of the RSC problem as an integer linear programming (ILP) problem. The key simplifying assumption (which may not be practical, but is instrumental for computing optimal solutions) is that the edge weights ω are in the form of $P, P^2, \cdots$ where $0 \leq P \leq 1$ and the exponents are arbitrary positive integers. We further simplify the ILP problem by assuming that the weights of all edges entering a node are the same.

We used CPLEX and Gurobi to solve the ILP problem and the exact solutions served as the baseline for comparing with the output of the `CGreedySurvivors` routine. As shown in Figure 5.5, for all of the cases tried (from 6 to 62 nodes) the output of our algorithm uses no more than 33% more robots than the optimal, and typically closer to 25%. This gives empirical justification for using the `CGreedySurvivors` routine as a fast, high quality approximate algorithm.

5.3.2 Application to search and rescue

We consider the search and rescue scenario from the introduction, using a subset of the storm data shown in Figure 5.1. The risk posed by storms to robots was analyzed by [42] using a model very similar to ours, where the probability of survival is the product of the probability of surviving each edge in a path. We refer interested readers to their paper, which includes a detailed weather and risk model.

We place sites in a uniform 15×15 grid and compute the edge weights by integrating the "base reflectivity" (the amount of radar energy reflected by the weather system) across the straight-line connection between sites. We set $p_s = 0.8$ and $H_j = 0.95$ for all nodes. Our routine takes an average of 20.3 seconds per path

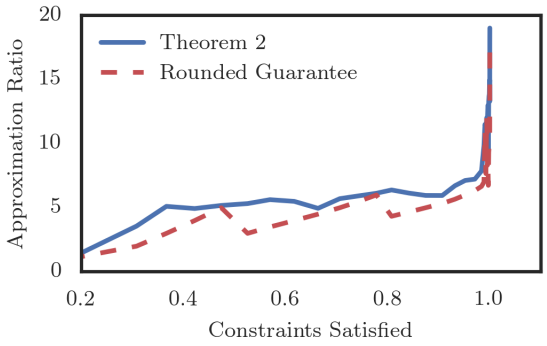


Figure 5.6: Bi-criteria performance comparison for the search and rescue scenario with 225 nodes and $p_v(j) = 0.95$. The top curve (blue) shows the approximation guarantee from Theorem 2. The lower dashed curve (red) shows the guarantee while enforcing integrality and monotonicity on the lower bound on the optimal team size from Theorem 2.

and finds a set of 36 paths which satisfy the coverage constraints exactly, but the guarantee from Theorem 2 gives an approximation ratio of 19.03. Using the bi-criteria guarantee and monotonicity of the lower bound we know that $|X_K^*| \geq 4$, which means the approximation ratio is at most 9. We can find tighter guarantees by relaxing the constraints: the first 27 robots visit 99.7% of the required expected number of nodes with approximation ratio 6.75, and the first 13 robots satisfy 80.7% of the constraints with approximation ratio 4.33. Figure 5.6 shows the fraction of constraints satisfied versus the approximation guarantee from Theorem 2 and a refined analysis which accounts for the integrality and monotonicity of the optimal solution. Hence, by slightly relaxing the constraint satisfaction one obtains reasonably tight approximation ratios.

5.3.3 Quality of cost-benefit approach

We implemented our algorithm using the Gurobi MILP solver. Orienteering problems are solved using the formulation from [30], using the built-in solver tolerance to set δ appropriately. We further improve run time by (1) using the cutoff parameter to terminate the search immediately if the solver determines there are no solutions in the region with better objective than $\hat{F}$ and (2) re-using the same solver model for each sub-problem which reduces overhead and allows the solver to re-use computation.

We compare our approach (with $\varepsilon = 0.1, \delta = 0.025$) against an exhaustive search on a cost-benefit path planning problem with objective $f(\rho) = \log(\sum_{j\in\rho} b_j) - \log(1 - \prod_{e\in\rho} \omega_e)$, with b_j uniformly distributed over [0,1] and ω_e uniformly distributed over [0.75,0.95]. We use random Delaunay graphs with 5 to 130 nodes, since Delaunay graphs (and the closely related Voronoi graph) have been applied to path planning and space discretizations, are easy to generate, and have $O(V)$ edges (which significantly reduces the run time of the exhaustive search).

We ran the exhaustive search for graphs up to 24 nodes (after which each solution took more than 100 seconds). A comparison of the run times averaged over 30 trials for each graph size is shown in Figure 5.8.

Note that the exact solver follows an $O(2^V)$ trend (as expected), while our approach appears to follow a $O(V^2)$ trend. This verifies our earlier claims that MILP solvers can often solve orienteering problems efficiently.

Furthermore our solutions are very high quality, as shown in Figure 5.9. In 78% of the trials where we have comparison data, our approach found the optimal solution, and in the remaining 22% the additive error was 0.014 on average and the relative error was 0.6% on average.

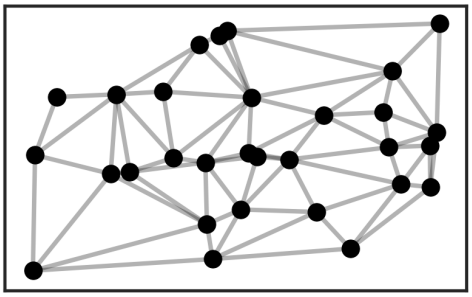


Figure 5.7: Example of a Delaunay graph with 30 nodes.

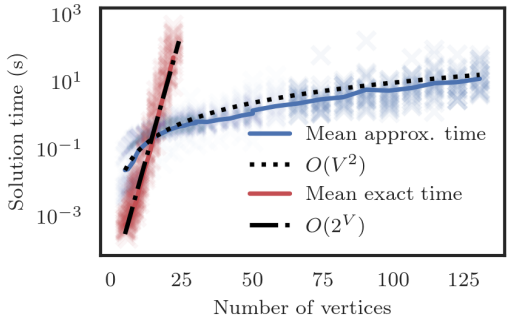


Figure 5.8: Runtime characterization of our algorithm (blue) versus an exact solver (red)

5.3.4 Comparison of uniform and non-uniform costs

We next compare the uniform and non-uniform cost variants for the search and rescue setting. Figure 5.10 shows a comparison of the cost/benefit ratio as a function of the fraction of constraints satisfied. The cost-benefit greedy approach performs significantly better than the greedy approach from [61], which highlights the importance of an algorithm which specifically optimizes the cost-benefit ration. The greedy algorithm solves the problem using 44 robots with expected cost of 8.94 robots, whereas the cost-benefit greedy approach uses a team of 15 robots with expected losses of 6.82 robots.

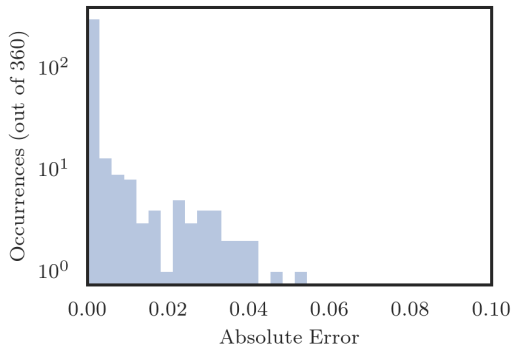


Figure 5.9: Frequency of absolute error values for threshold $\varepsilon = 0.1$ and $\delta = 0.025$. Note that $f(X^*) \simeq 2$ for these trials. Data is for graphs with up to 24 nodes.

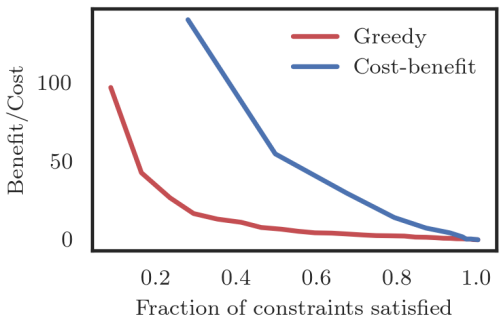


Figure 5.10: Comparison of performance of the greedy algorithm versus the cost-benefit greedy algorithm in terms of cost.

5.4 Conclusion

We consider the *Risk-Sensitive Coverage* problem, where we seek the smallest set of routes which satisfy visit probability thresholds and survival probability thresholds. We demonstrate that the RSC is an instance of the submodular set cover problem with a very large ground set. We give an approximate cost-benefit greedy routine for constructing a feasible solution to the RSC, and then provide a bi-criteria approximation guarantee which ensures that the solution returned is close to an optimal solution of a similar problem. We then compare the solutions returned by our routine to an exact solution for a special case of the RSC, and demonstrate the applicability of the approach on simulated data which represents a search and rescue setting during a severe storm.

Our algorithm for the cost-benefit path planning problem implicitly searches the Pareto optimal set by solving a sequence of orienteering problems. We derive convergence bounds on the absolute and relative error, and give empirical evidence of the empirical quality and efficiency of our approach. We stress that, although we use a scalarization approach to the multi-objective cost-benefit planning problem, our approach can handle very *non-linear* combinations of edge and node weights such as the cost benefit ratio from the RIPP, $\sum_{j\in\rho} b_j/1-\prod_{e\in\rho} c_e$, the cost benefit ratio from the path exploration problem, $\prod_{j\in\rho} b_j/\sum_{e\in\rho} c_e$, and the upper confidence bounds from reinforcement learning, $\sum_{j\in\rho} \mu_j + \sqrt{\sum_{j\in\rho} \sigma_j^2}$.

Chapter 6

Conclusions

Coordinating teams of robots to complete sophisticated missions in risky environments is an important and challenging problem which has few existing solutions. This thesis considers a 'risky traversal' model where each time a robot travels between a pair of destinations, it may fail catastrophically and not complete its mission. This model is not compatible with existing approaches, and this thesis provides a set of tools based on submodular optimization for solving the problems of (1) maximizing expected rewards collected by a team, (2) updating the paths in response to information gathered on-line, (3) ensuring that the set of paths satisfies complex independence constraints represented using a matroid, and (4) guaranteeing coverage in dangerous environments while minimizing the resources required.

Our tools are based on variations of the greedy algorithm, which means our approaches scale linearly as the team size grows, and linearly as the number of robot types available grows. To solve the greedy sub-problem we rely on an oracle routine for the well known orienteering problem, which has a variety of practical solution approaches despite being NP-hard. Using the submodular structure of risky traversal objectives, our approaches have bounded suboptimality, often guaranteeing our solution achieves at least $\simeq 60\%$ of the optimal solution. This work provides a foundation for further studies on the structure of risky traversal problems and design of efficient solution algorithms.

6.1 Summary

In Chapter 3, we study the Team Surviving Orienteers problem, where the goal is to maximize the expected rewards that a team of fixed size collects. We provide a linearization approach which approximates the non-linear and history dependent greedy sub-problem using a standard orienteering problem. Our algorithm has a $1-e^{-p_s/\lambda}$ guarantee, and complexity grows at worst linearly in the number of robots and types of robots available. We also study an on-line variant of the TSO problem, where paths are adapted in response to survival/failure events, and the heterogeneous TSO problem, where we optimize simultaneously over the team composition and paths taken.

In Chapter 4, we consider independence constraints represented using a matroid, which generalizes linear independence for set functions. We demonstrate how to represent such constraints efficiently using sub-graph decompositions and give a greedy algorithm with a $p_s/(p_s+\lambda)$ suboptimality bound. We then describe a continuous greedy algorithm, which performs gradient ascent over the multilinear extension of the objective function. We show that the continuous greedy algorithm with discretization step size $\delta \ll 1$ has a $(1-\delta)(1-e^{-p_s/(\lambda+\delta)})$ guarantee, which matches the results from Chapter 3 as δ becomes small.

In Chapter 5, we formalize the Risk Sensitive Coverage problem, which is the dual to the TSO problem. We show that the RSC problem is an instance of the submodular set cover, and can be approximated efficiently with a cost-benefit greedy algorithm. We give a general algorithm for solving cost-benefit greedy path planning problems and provide convergence rate bounds for both relative and absolute error. We then provide an algorithm with a bi-criteria guarantee on its suboptimality relative to the optimal solution to a closely related problem. Specifically, if the cost-benefit sub-problem is solved within factor $\alpha \leq 1$, then the cost of the first L paths our algorithm selects is at most $\frac{1}{\alpha}(1+\log(\Delta_L/\alpha))$ and satisfies at least fraction L/K of the constraints, where Δ_L is the ratio of benefit for the first and Lth path selected.

6.2 Future Directions

The models used in this thesis have a number of limitations which future work should address. We primarily focus on environments with well understood reward structure, constraints, and survival probability distributions. Using the foundations developed in this thesis we hope future research develops in the following areas:

Alternate approaches to the greedy sub-problem In this work we approach the challenge of planning with risky traversal by noting that p_s is close to 1, and so by linearizing the problem using the fact that survival probabilities are in $[p_s, 1]$ we get a fairly tight guarantee. However in some scenarios (such as the monitoring problem presented in Section 4.6.1) some regions are very difficult to reach but yield significant benefits. In this work we handle this case using a matroid constraint to relax p_s, but this loosens the guarantees we can provide a-priori. Future work should consider alternate approaches to solving the greedy sub-problem. One option is to solve a submodular orienteering problem directly, however this ignores the specific structure of our problem and is computationally intensive. One could also consider solving multiple orienteering problems with different thresholds, choosing the best result (along the lines of the cost-benefit path planning problem). This would preserve the same order of complexity and may give more flexibility on the guarantees available.

More sophisticated models The results in this work rely on an independence assumption for node rewards (i.e., reward gathered at node j does not change reward available at node j'), on survival probability events for a single robot (i.e., surviving edge e is independent of surviving edge e'), and survival events for different

robots (i.e. robot k surviving edge e is independent of robot k' surviving edge e). While this leads to simplified analysis and is supported by other work in the field [42], these assumptions limit the usefulness of this model in real-world scenarios. Some work on 'correlated orienteering problems' [72], Monte Carlo planning methods [73], or Markov decision processes [74] may help lift these assumptions. Provided that the greedy sub-problems remain approximable under these models, the main results of this thesis should still hold.

Planning with uncertain survival probabilities While our work can handle uncertain node values or node presence, planning with unknown survival probabilities poses a challenge because measurements can change which paths are deemed 'feasible'. While one could impose unnatural constraints on learning such as a negative monotone constraint on the estimated survival probability of an edge (meaning information can only make one more pessimistic about survival probabilities), this is difficult to justify in practice. Future research should consider alternate ways of formulating risky traversal problems which can naturally incorporate uncertain edge probabilities into the optimization process.

Extensions to policies One way to handle uncertain edge probabilities is by planning over *policies*, which are much richer actions than the paths which are used in this work. A policy encodes the response to new information learned on-line, and so the set of feasible policies does not grow by learning that an edge is safer than originally expected. Reasoning over policies is also interesting because it is a very natural way to plan in uncertain environments, could allow for parallel exploration in unknown graphs, and can be extended to game theoretic formulations with non-cooperative agents. The principle challenge with a policy based approach is that the cardinality of the ground set is much, much larger than the ground set of feasible paths. While there are some approaches for finding policies to, e.g. partially observable Markov decision processes, they do not scale well as the complexity of the problem grows. As policy optimization becomes better understood, a submodular optimization perspective is likely to help by reducing the optimization space to a policy for a single robot rather than for the entire team.

Cooperation with human operators The big-picture goal of this work is to understand coordination in risky environments thoroughly in order to facilitate missions with human operators, a crucial step in order for robots to live up to their promise to do "dull, dirty, and *dangerous*" work. In particular missions such as search and rescue in buildings (e.g. during a fire) or damage assessment during a natural disaster require planning approaches which can simultaneously handle danger to the robot, coordination with other robots, coordination with human operators, and exploring unknown environments. While many of these topics are covered in this dissertation, there is a significant amount of work yet to do before these approaches can be deployed. Namely, incorporating human robot *interaction* with more sophisticated robot models while retaining efficient solutions will require significant effort.

Appendix A

Technical Result on Poisson Binomial Distributions

In the following we prove the technical lemma stated in the background section. We start the proof by considering a sequence of Poisson binomial distributions $f_0, f_1, \ldots$. The parameters of the nth distribution are denoted as $\{p_{n,k}\}_{k=1}^{K}$, with $p_{n,1} \leq p_{n,2} \leq \cdots \leq p_{n,K}$. The parameters of the $n+1$st distribution are

$$\{p_{n+1,k}\}_{k=1}^{K} = \left\{ \frac{p_{n,1}+p_{n,K}}{2}, \{p_{n,k}\}_{k=2}^{K-1}, \frac{p_{n,1}+p_{n,K}}{2} \right\},$$

that is, the largest and smallest event probabilities of the nth distribution are averaged to form the $n+1$st distribution. Note that we re-sort the parameters after constructing them from the nth distribution, so it is still true that $p_{n+1,j} \leq p_{n+1,k}$ for $j \leq k$.

It is easy to verify that this sequence converges to the binomial distribution with parameters K and $p = \frac{1}{K}\sum_{k=1}^{K} p_k$. We are interested in showing that the tails of the sequence become heavier as n increases. We begin by making some basic observations:

Lemma 15. *Define*

$$\varepsilon_n := \frac{1}{2}(p_{n,K} - p_{n,1})$$

and

$$\bar{p}_n := \frac{1}{2}(p_{n,K} + p_{n,1}).$$

Then

$$p_{n,1}p_{n,K} = \bar{p}_n^2 - \varepsilon_n^2,$$

$$(1-p_{n,1})(1-p_{n,K}) = (1-\bar{p}_n)^2 - \varepsilon_n^2,$$

and

$$p_{n,1}(1-p_{n,K})+p_{n,K}(1-p_{n,1}) = 2\bar{p}_n(1-\bar{p}_n)+2\varepsilon_n^2.$$

Proof. Each of these statements follows from straightforward algebra:

$$\begin{aligned}\bar{p}_n^2-\varepsilon_n^2 &= \frac{1}{4}\left(p_{n,1}^2+2p_{n,1}p_{n,K}+p_{n,K}^2\right)\\ &\quad -\frac{1}{4}\left((p_{n,K}^2-2p_{n,1}p_{n,K}+p_{n,1}^2)\right)\\ &= \frac{1}{4}(4p_{n,1}p_{n,K}) = p_{n,1}p_{n,K}\end{aligned}$$

$$\begin{aligned}(1-\bar{p}_n)^2-\varepsilon_n^2 &= 1-2\bar{p}_n+\bar{p}_n^2-\varepsilon_n^2\\ &= 1-(p_{n,1}+p_{n,K})+p_{n,1}p_{n,K}\\ &= (1-p_{n,1})(1-p_{n,K})\end{aligned}$$

$$\begin{aligned}2\bar{p}_n(1-\bar{p}_n)+2\varepsilon_n^2 &= -2(\bar{p}_n^2-\varepsilon_n^2)+2\bar{p}_n\\ &= -2(p_{n,1}p_{n,K})+(p_{n,1}+p_{n,K})\\ &= p_{n,1}(1-p_{n,K})+pn,K(1-p_{n,1})\end{aligned}$$

□

It is useful to define an auxiliary sequence of Poisson binomial probability mass functions

$$g_n(m) = f(m;\{p_{n,k}\}_{k=2}^{K-1}),$$

which correspond to the probabilities of m successes *excluding* the most and least likely "events" of the nth distribution. Note that by definition $g_n(m) = 0$ if $m < 0$ or $m > K-2$. We also define notation for the first and second-order finite difference of $g_n(m)$, which are crucial quantities in our inequalities below.

$$\begin{aligned}\Delta_{1,n}(m) &:= g_n(m)-g_n(m-1),\\ &\text{and}\\ \Delta_{2,n}(m) &:= \Delta_{1,n}(m)-\Delta_{1,n}(m-1)\\ &= g_n(m)-2g_n(m-1)+g_n(m-2).\end{aligned}$$

Using these relationships, we can form a succinct recursive description of $f_n(m)$:

Lemma 16. *For the sequence of probability mass functions above, we have*

$$f_n(m) = f_{n+1}(m)-\varepsilon_n^2\Delta_{2,n}(m),$$

and for $F_n(m') := \sum_{m=0}^{m'} f_n(m)$,

$$F_n(m') = F_{n+1}(m') - \varepsilon_n^2 \Delta_{1,n}(m').$$

Proof. By definition of the probability mass function and $g_n(m)$,

$$\begin{aligned}
f_n(m) &= p_{n,1}p_{n,K}g_n(m) \\
&\quad + (p_{n,1}(1-p_{n,K}) + p_{n,K}(1-p_{n,1}))g_n(m-1) \\
&\quad + (1-p_{n,1})(1-p_{n,K})g_n(m-2) \\
&= \bar{p}_n^2 g_n(m) + \bar{p}_n(1-\bar{p}_n)g_n(m-1) + (1-\bar{p}_n)^2 g_n(m-2) \\
&\quad - \varepsilon_n^2(g_n(m) - 2g_n(m-1) + g_n(m-2)) \\
&= f_{n+1}(m) - \varepsilon_n^2 \Delta_{2,n}(m).
\end{aligned}$$

The second equality follows from the identities in Lemma 15, and the second equality follows by definition of $f_{n+1}(m)$ and $\Delta_{2,n}$.

Now taking the summation gives us the second statement:

$$\begin{aligned}
F_n(m') &= \sum_{m=0}^{m'} f_{n+1}(m) - \varepsilon_n^2(g_n(m) - 2g_n(m-1) + g_n(m-2)) \\
&= F_{n+1}(m') - \varepsilon_n^2 \left(\sum_{m=0}^{m'} g_n(m) - 2\sum_{m=0}^{m'-1} g_n(m) + \sum_{m=0}^{m'-2} g_n(m) \right) \\
&= F_{n+1}(m') - \varepsilon_n^2(g_n(m') - g_n(m'-1)) \\
&= F_{n+1}(m') - \varepsilon_n^2 \Delta_{1,n}(m').
\end{aligned}$$

□

This lemma gives us an *exact* characterization of the difference between successive distributions in our sequence. Specifically, $f_n(m) \leq f_{n+1}(m)$ if and only if the second order finite difference, $\Delta_{2,n}(m)$, is non-negative, and $F_n(m') \leq F_{n+1}(m')$ if and only if the first order finite difference, $\Delta_{1,n}(m')$ is non-negative. In the following lemma, we give a sufficient condition on m to ensure that $\Delta_{1,n}(m) \geq 0$.

Lemma 17. *Let* $\{p_{n,k}\}_{k=1}^K$ *and* μ *be defined as in Lemma 1, and* $\Delta_{1,n}(m)$ *defined as the first order finite difference of* $g_n(m)$*, the Poisson binomial distribution with parameters* $\{p_{n,k}\}_{k=2}^{K-1}$*. Then for* $m \leq (1-p_{1,K})(K-2)\frac{\mu}{1-\mu} + p_{1,K}$, $\Delta_{1,n}(m) \geq 0$.

Proof. We start by expressing $g_n(m)$ using the recursive characterization of the Poisson binomial probability mass function given by [75]:

$$g_n(m) = \frac{1}{m}\sum_{i=1}^{m}(-1)^{i-1} g_n(m-i)T_n(i),$$

where $T_n(i) = \sum_{k=2}^{K-1}\left(\frac{p_{n,k}}{1-p_{n,k}}\right)^i$. Note that for $i \geq 2$, we have $T_n(i) \leq T_n(i-1)\frac{p_{1,K}}{1-p_{1,K}}$. The case $\Delta_{1,n}(m+1) \geq 0$

is equivalent to saying that $\frac{g_n(m+1)}{g_n(m)} \geq 1$. Using the recursive expression above,

$$\begin{aligned}
\frac{g_n(m+1)}{g_n(m)} &= \frac{T_n(1)}{m+1} - \frac{\sum_{i=1}^{m}(-1)^{i-1} g_n(m-i) T_n(i+1)}{(m+1) g_n(m)} \\
&\geq \frac{T_n(1)}{m+1} - \frac{\sum_{i=1}^{m}(-1)^{i-1} g_n(m-i) T_n(i) \left(\frac{p_{1,K}}{1-p_{1,K}}\right)}{(m+1) g_n(m)} \\
&\geq \frac{T_n(1)}{m+1} - \frac{m}{m+1}\left(\frac{p_{1,K}}{1-p_{1,K}}\right) \\
&\geq \frac{K-2}{m+1}\frac{\mu}{1-\mu} - \frac{m}{m+1}\frac{p_{1,K}}{1-p_{1,K}}.
\end{aligned}$$

Solving for $m+1$ we have that $\Delta_{1,n}(m+1) \geq 0$ if

$$m+1 \leq (1-p_{1,K})\left((K-2)\frac{\mu}{1-\mu}\right) + p_{1,K}$$

□

Combining Lemmas 16 and 17 completes the proof for Lemma 1.

Appendix B

Guarantees for the AGCA

Derivation of the objective function for the ACGA

In this section we derive the equivalent form of the objective function for the continuous greedy algorithm given in Lemma 11:

$$F(y+\delta \mathbf{1}_\rho) - F(y) = \delta \sum_{j=1}^{V} d_j \frac{\mathbb{E}[z_j^\ell(\rho)]}{1-y_\rho \mathbb{E}[z_j^\ell(\rho)]} \mathbb{E}[p_j(0, R(y))].$$

At any point in the algorithm $y_\rho < 1$ if ρ is a candidate solution to the greedy sub-problem. We begin by giving three useful identities about the distribution of $R(y)$, then express the objective function in terms of $\mathbb{E}[\Delta f(\rho \mid R(y))]$ (a useful result for subsequent proofs), and finally derive the statement given in Lemma 11.

1. Let $P_y(X)$ be the probability that $R(y) = X$. Then

$$P_{y+\delta \mathbf{1}_\rho}(X) = P_y(X)\left(1+\delta\left(\frac{\mathbb{I}\{\rho \in X\}}{y_\rho} - \frac{\mathbb{I}\{\rho \notin X\}}{1-y_\rho}\right)\right) \tag{B.1}$$

   which follows directly from the definition of R.

2. For any X, ρ,

$$P_y(X \cup \rho) = P_y(X \setminus \rho)\frac{y_\rho}{1-y_\rho}. \tag{B.2}$$

3. By definition of p_j, we have for all X, ρ:

$$p_j(0, X \cup \rho) = p_j(0, X \setminus \rho)(1 - \mathbb{E}[z_j(\rho)]),$$

which along with Equation B.2 gives

$$p_j(0,X\setminus\rho)P_y(X\setminus\rho)+p_j(0,X\cup\rho)P_y(X\cup\rho)=p_j(0,X\setminus\rho)P_y(X\setminus\rho)\left(1+\frac{y_\rho(1-\mathbb{E}[z_j(\rho)])}{1-y_\rho}\right),$$

which implies the third identity:

$$\mathbb{E}[p_j(0,R(y))]=\frac{1-y_\rho\mathbb{E}[z_j(\rho)]}{1-y_\rho}\sum_{X\subseteq\mathscr{X}\setminus\rho}p_j(0,X)P_y(X) \tag{B.3}$$

Now using these identities with the definition of F, we have

$$\begin{aligned}
F(y+\delta\mathbf{1}_\rho)-F(y)&=\sum_{X\subseteq\mathscr{X}}f(x)\left(P_{y+\delta}(X)-P_y(X)\right) && \text{By definition of } F\\
&=\sum_{X\subseteq\mathscr{X}}\delta f(X)P_y(X)\left(\frac{\mathbb{I}\{\rho\in X\}}{y_\rho}-\frac{\mathbb{I}\{\rho\notin X\}}{1-y_\rho}\right) && \text{Using eq. B.1}\\
&=\delta\sum_{X\subseteq\mathscr{X}\setminus\rho}P_y(X)\left(\frac{f(X\cup\rho)}{y_\rho}\frac{y_\rho}{1-y_\rho}-\frac{f(X)}{1-y_\rho}\right) && \text{Using eq. B.2}\\
&=\frac{\delta}{1-y_\rho}\sum_{X\subseteq\mathscr{X}\setminus\rho}\Delta f(\rho\mid X)P_y(X) && \text{By definition of } \Delta f
\end{aligned}$$

This result combined with Equation B.3 and the fact that $\Delta f(\rho\mid X\cup\rho)=0$ gives the two desired results:

$$F(y+\delta\mathbf{1}_\rho)-F(y)=\frac{\delta}{1-y_\rho}\mathbb{E}[\Delta f(\rho\mid R(y))] \tag{B.4}$$

$$=\frac{\delta}{1-y_\rho}\sum_{j=1}^{V}d_j\mathbb{E}[z_j(\rho)]\sum_{X\subseteq\mathscr{X}\setminus\rho}p_j(0,X)P_y(X) \qquad \text{By definition of } f \tag{B.5}$$

$$=\delta\sum_{j=1}^{V}d_j\frac{\mathbb{E}[z_j(\rho)]}{1-y_\rho\mathbb{E}[z_j(\rho)]}\mathbb{E}[p_j(0,R(y))] \qquad \text{Applying eq. B.3} \tag{B.6}$$

Performance guarantee for the ACGA

In this section we provide the proof for the statement of Theorem 7, which we repeat below:

Let X^* be an optimal solution to the MTSO problem and $\hat{X}$ be the output of the `MCGreedySurvivors` routine with parameters δ and λ. Then the value of the set $\hat{X}$ is lower bounded by a constant factor of the optimum:

$$f(\hat{X})\geq f(X^*)(1-\delta)\left(1-\exp\left(-\frac{p_s}{\lambda+\delta p_s}\right)\right).$$

Outline of argument – The proof follows the argument of [17] closely. We begin by lower bounding the increase in $F(y)$ between subsequent steps and then use a recursive argument to bound the value of $F(y)$ after

the last iteration. Since this section deals primarily with the evolution of y, we denote the state of y after the ℓth iteration of the ith step as $y(i,\ell)$, and use the shorthand $y(i,K) = y(i) = y(i+1,0)$. We also denote the path selected by the algorithm during the ℓth iteration of the ith step by $\rho_{i,\ell}$.

Note on feasibility – An important consequence of the exchange properties of matroids is that during any step i there is an ordering of the elements in an optimal set $\rho_1^*, \ldots, \rho_K^*$ such that ρ_ℓ^* is a candidate solution when solving the sub-problem during the ℓth iteration. This means we can combine Lemmas 11 and 10 to bound the increase in $F(y)$ between steps.

Note on the sub-problem – When solving the sub-problem, we consider the paths with $y_\rho > 0$ explicitly and solve the greedy sub-problem using the approximation $y_\rho = 0$. Since the node reward is an increasing function of y_ρ, we still have a λ–approximate guarantee for the sub-problem:

$$\rho \in \underset{\rho \in \mathscr{X}_F(X(i),\mathscr{I})}{\arg\max} \sum_{j=1}^{V} d_j \frac{\hat{v}(j)}{1 - y_\rho \mathbb{E}[z_j(\rho)]},$$

which means we can apply Lemmas 10 and 11 to guarantee that $\rho_{i,\ell}$ is within a multiplicative factor of p_s/λ of the optimal.

Now we can bound $F(y(i+1)) - F(y(i))$ below.

$$\begin{aligned}
F(y(i+1)) - F(y(i)) &= \sum_{\ell=1}^{K} F(y(i,\ell)) - F(y(i,\ell-1)) = \delta \sum_{\ell=1}^{K} \mathbb{E}[\Delta f(\rho_{i,\ell} \mid R(y))] && \text{Telescoping sum} \\
&\geq \delta \frac{p_s}{\lambda} \sum_{\ell=1}^{K} \mathbb{E}[\Delta f(\rho_\ell^* \mid R(y(i,\ell-1)))] && \text{Lemma 10, } \rho_\ell^* \text{ feasible} \\
&\geq \frac{\delta p_s}{\lambda} \left(\mathbb{E}[f(X^* \cup R(y(i)))] - \mathbb{E}[f(R(y(i+1)))]\right) && \text{Submodularity} \\
&\geq f(X^*) \frac{\delta p_s}{\lambda} - \frac{\delta p_s}{\lambda} F(y(i+1)) && \text{Monotonicity}
\end{aligned}$$

Now we can rearrange the inequality above to get

$$(F(y(i+1)) - f(X^*)) \left(1 + \frac{\delta p_s}{\lambda}\right) \geq (F(y(i)) - f(X^*)), \tag{B.7}$$

which applied recursively gives

$$\begin{aligned}
F(y(1/\delta)) &\geq f(X^*) \left(1 - \left(1 + \frac{\delta p_s}{\lambda}\right)^{-1/\delta}\right) \\
&\geq f(X^*) \left(1 - \exp\left(-\frac{p_s}{\lambda + p_s \delta}\right)\right).
\end{aligned}$$

The last inequality is because $1 + x \leq e^x$ implies $(1+x)^{-N} \leq e^{-Nx/(1+x)}$. The theorem statement follows since we enforce that the rounded result $\hat{X}$ satisfies $f(\hat{X}) \geq (1-\delta) F(y(1/\delta))$.

Correctness for `UpdateWeights` routine

We introduce some new notation to analyze the `UpdateWeights` routine. Given a vector y, let ρ_n be the nth non-zero coordinate of y, and define $R^\ell(y) := R(y) \cap \{\rho_n\}_{n=1}^{\ell}$ as the random set restricted to the first ℓ nonzero coordinates. Now define

$$w_m^\ell := \mathbb{E}[p_j(0, R^\ell(y)) \mathbb{I}\{|R^\ell(y)| = m\}],$$

which is the expected probability that zero of the paths in $R^\ell(y)$ visit node j and there are m paths in $R^\ell(y)$. If M is the number of non-zero entries of y, then by the definition of the expectation we have,

$$\mathbb{E}[p_j(0, R(y))] = sum_{m=0}^{M} w_m^M,$$

since $R(y)$ must have between 0 and M elements.

For $\ell = 0$, the weights w_m^ℓ are zero unless $m = 0$. For $\ell > 0$ we can use the product form of $p_j(0, X)$ to express w_m^ℓ in terms of $w_{m-1}^{\ell-1}$ and $w_m^{\ell-1}$:

$$w_m^\ell = w_{m-1}^{\ell-1} \left((1 - \mathbb{E}[z_j^\ell(\rho_\ell)]) y_{\rho_\ell} \right) + w_m^{\ell-1} \left(1 - y_{\rho_\ell}\right).$$

This expression has an intuitive meaning, as it is the two ways that $R^\ell(y)$ can have m elements: either $y_\ell \in R^\ell(y)$ and $|R^{\ell-1}(y)| = m-1$, or $y_\ell \notin R^\ell(y)$ and $|R^{\ell-1}(y)| = m$. In both cases we update the weights and probability of the event occurring by multiplying by the appropriate coefficients.

The `UpdateWeights` routine simply applies this iterative approach to compute $\{w_m^M\}_{m=0}^M$ and then sums the weights to find $\mathbb{E}[p_j(0, R(y))]$.

Expected complexity of `SwapRounding`

Let p_f be the probability that `SwapRounding` fails to return a satisfactory result. Then the expected number of calls is

$$\sum_{n=1}^{\infty} n p_f^{n-1}(1 - p_f) = \frac{1}{1 - p_f}.$$

Now using the bound from [25], $p_f \leq \exp(-F(y)\delta^2/8) \leq \exp(-p_s\delta^2/8)$, and from the inequality $e^{-x} \leq 1 - x + x^2/2$ we get $\frac{1}{1-e^{-x}} \leq \frac{2}{2x-x^2}$. Combining these inequalities gives the cited result.

Note that we could make the while loop terminate after δ^{-2} iterations which would make the statement of Theorem 7 hold with probability at least $1 - e^{-p_s/8}$, and the complexity be deterministic and $O(K^2\delta^{-3})$.

Bibliography

[1] A. Cros, N. Ahamad Fatan, A. White, S. Teoh, S. Tan, C. Handayani, C. Huang, N. Peterson, R. Venegas Li, H. Y. Siry, R. Fitriana, J. Gove, T. Acoba, M. Knight, R. Acosta, N. Andrew, and D. Beare, "The Coral Triangle Atlas: An integrated online spatial database system for improving coral reef management," *PLoS ONE*, vol. 9, no. 6, pp. 1–7, 2014.

[2] International Chamber of Commerce: Commercial Crime Services. (2017) IMB piracy reporting centre. Available at https://www.icc-ccs.org/piracy-reporting-centre.

[3] NOAA National Weather Service Radar Operations Center, "NOAA next generation radar (NEXRAD) level II base data," NOAA National Centers for Environmental Information, 1991.

[4] A. Gunawan, H. Lau, and P. Vansteenwegen, "Orienteering problem: A survey of recent variants, solution approaches and applications," *European Journal of Operational Research*, vol. 255, no. 2, pp. 315–332, 2016.

[5] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011.

[6] C. Chekuri and M. Pál, "A recursive greedy algorithm for walks in directed graphs," in *IEEE Symp. on Foundations of Computer Science*, 2005.

[7] H. Zhang and Y. Vorobeychik, "Submodular optimization with routing constraints," in *Proc. AAAI Conf. on Artificial Intelligence*, 2016.

[8] I.-M. Chao, B. Golden, and E. Wasil, "The team orienteering problem," *European Journal of Operational Research*, vol. 88, no. 3, pp. 464–474, 1996.

[9] A. M. Campbell, M. Gendreau, and B. W. Thomas, "The orienteering problem with stochastic travel and service times," *Annals of Operations Research*, vol. 186, no. 1, pp. 61–81, 2011.

[10] A. Paul, D. Freund, A. Ferber, D. B. Shmoys, and D. Williams, "Prize-collecting TSP with a budget constraint," in *LIPIcs-Leibniz Int. Proc. in Informatics*, 2017.

[11] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1–11, 2013.

[12] H. N. Psaraftis, M. Wen, and C. A. Kontovas, "Dynamic vehicle routing problems: Three decades and counting," *Networks*, vol. 67, no. 1, pp. 3–31, 2016.

[13] *Gurobi Optimizer reference manual*, Gurobi Optimization, Inc., 2016.

[14] *ILOG CPLEX User's guide*, IBM ILOG, 1987.

[15] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, vol. 34, pp. 707–755, 2009.

[16] N. Atanasov, J. Le Ny, K. Daniilidis, and G. Pappas, "Decentralized active information aquisition: Theory and application to multi-robot SLAM," in *Proc. IEEE Conf. on Robotics and Automation*, 2015.

[17] A. Badanidiyuru and J. Vondrák, "Fast algorithms for maximizing submodular functions," in *ACM-SIAM Symp. on Discrete Algorithms*, 2014.

[18] A. Krause and D. Golovin, "Submodular function maximization," in *Tractability: Practical approaches to hard problems*. Cambridge Univ. Press, 2014.

[19] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*, 1st ed., R. Graham, B. Korte, L. Lov'asz, A. Wigderson, and G. Ziegler, Eds. Springer Science & Business Media, 2002.

[20] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions–I," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[21] K. Wei, R. Iyer, and J. Bilmes, "Fast multi-stage submodular maximization," in *Int. Conf. on Machine Learning*, 2014.

[22] U. Feige, "A threshold of ln n for approximating set cover," *Journal of the Association for Computing Machinery*, vol. 45, no. 4, pp. 634–652, 1998.

[23] M. Fisher, G. Nemhauser, and L. Wolsey, "An analysis of approximations for maximizing submodular set functions –II," in *Polyhedral Combinatorics*. Springer, 1978.

[24] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a submodular set function subject to a matroid constraint," in *Int. Conf. on Integer Programming and Combinatorial Optimization*, 2007.

[25] C. Chekuri, J. Vondrák, and R. Zenklusen, "Dependent randomized rounding via exchange properties of combinatorial structures," in *IEEE Symp. on Foundations of Computer Science*, 2010.

[26] L. Wolsey, "An analysis of the greedy algorithm for the submodular set covering problem," *Combinatorica*, vol. 2, no. 4, pp. 385–393, 1982.

[27] B. L. Golden, L. Levy, and R. Vohra, "The orienteering problem," *Naval Research Logistics*, vol. 34, no. 3, pp. 307–318, 1987.

[28] K. Chen and S. Har-Peled, "The orienteering problem in the plane revisited," in *ACM Symp. on Computational Geometry*, 2006.

[29] C. Chekuri, N. Korula, and M. Pál, "Improved algorithms for orienteering and related problems," *ACM Transactions on Algorithms*, vol. 8, no. 3, pp. 23:1–23:27, 2012.

[30] I. Kara, P. Biçakci, and T. Derya, "New formulations for the orienteering problem," *Procedia Economics and Finance*, vol. 39, pp. 849–854, 2016.

[31] S. Wagner and M. Affenzeller, "HeuristicLab: A generic and extensible optimization environment," in *Adaptive and Natural Computing Algorithms*. Springer, 2005.

[32] P. Vansteenwegen, W. Souffriau, G. V. Berghe, and D. Van Oudheusden, "Iterated local search for the team orienteering problem with time windows," *Computers & Operations Research*, vol. 36, no. 12, pp. 3281–3290, 2009.

[33] A. Gupta, R. Krishnaswamy, V. Nagarajan, and R. Ravi, "Approximation algorithms for stochastic orienteering," in *ACM-SIAM Symp. on Discrete Algorithms*, 2012.

[34] P. Varakantham and A. Kumar, "Optimization approaches for solving chance constrained stocahstic orienteering problems," in *Proc. Int. Conf. on Algorithmic Decision Theory*, 2013.

[35] G. Laporte, F. Louveaux, and H. Mercure, "Models and exact solutions for a class of stochastic location-routing problems," *European Journal of Operational Research*, vol. 39, no. 1, pp. 71–78, 1989.

[36] B. Golden and J. Yee, "A framework for probabilistic vehicle routing," *AIIE Transactions*, vol. 11, no. 2, pp. 109–112, 1979.

[37] W. Stewart and B. Golden, "Stochastic vehicle routing: A comprehensive approach," *European Journal of Operational Research*, vol. 14, no. 4, pp. 371–385, 1983.

[38] G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," *Int. Journal of Robotics Research*, vol. 33, no. 9, pp. 1271–1287, 2014.

[39] S. Jorgensen, R. Chen, M. Milam, and M. Pavone, "The team surviving orienteers problem: Routing robots in uncertain environments with survival constraints," in *IEEE Int. Conf. on Robotic Computing*, 2017.

[40] ——, "The team surviving orienteers problem: Routing teams of robots in uncertain environments with survival constraints," *Autonomous Robots*, vol. 42, no. 4, pp. 927–952, 2018.

[41] O. Vaněk, M. Jakob, O. Hrstka, and M. Pěchouček, "Agent-based model of maritime traffic in piracy affected waters," *Transportation Research Part C: Emerging Technologies*, vol. 36, pp. 157–176, 2013.

[42] B. Zhang, L. Tang, and M. Roemer, "Probabilistic planning and risk evaluation based on ensemble weather forecasting," *IEEE Transactions on Automation Sciences and Engineering*, vol. PP, no. 99, pp. 1–11, 2017.

[43] J. Haldane, "A note on inverse probability," *Mathematical Proc. of the Cambridge Philosophical Society*, vol. 28, no. 1, pp. 55–61, 1932.

[44] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, and N. Vathis, "Approximation algorithms for the arc orienteering problem," *Information Processing Letters*, vol. 115, no. 2, pp. 313–315, 2015.

[45] A. Bock and L. Sanità, "The capacitated orienteering problem," *Discrete Applied Mathematics*, vol. 195, no. C, pp. 31–42, 2015.

[46] N. A. Lynch, *Distributed Algorithms*. Morgan Kaufmann, 1996.

[47] R. Smith, M. Schwager, S. Smith, B. Jones, D. Rus, and G. Sukhatme, "Persistent ocean monitoring with underwater gliders: Adapting sampling resolution," *Journal of Field Robotics*, vol. 28, no. 5, pp. 714–741, 2011.

[48] Ç. Koç, T. Bektaş, O. Jabali, and G. Laporte, "Thirty years of heterogeneous vehicle routing," *European Journal of Operational Research*, vol. 249, no. 1, pp. 1–21, 2016.

[49] R. Lahyani, M. Khemakhem, and F. Semet, "Rich vehicle routing problems: From a taxonomy to a definition," *European Journal of Operational Research*, vol. 241, no. 1, pp. 1–14, 2015.

[50] A. Hoff, H. Andersson, M. Christiansen, G. Hasle, and A. Løkketangen, "Industrial aspects and literature survey: Fleet composition and routing," *Computers & Operations Research*, vol. 37, no. 12, pp. 2041–2061, 2010.

[51] E. Kelareva, K. Tierney, and P. Kilby, "CP methods for scheduling and routing with time-dependent task costs," *EURO Journal on Computational Optimization*, vol. 2, no. 3, pp. 147–194, 2014.

[52] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[53] S. Smith, J. Tůmová, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal-logic constraints," *Int. Journal of Robotics Research*, vol. 30, no. 14, pp. 1695–1708, 2011.

[54] A. Ulusoy, S. Smith, and C. Belta, "Optimal multi-robot path planning with ltl constraints: Guaranteeing correctness through synchronization," in *Int. Symp. on Distributed Autonomous Robotic Systems*, 2014.

[55] K. Murota, *Matrices and Matroids for Systems Analysis*, 1st ed. Springer Science & Business Media, 2009.

[56] S. Jawaid and S. Smith, "Informative path planning as a maximum travelling salesman problem with submodular rewards," *Discrete Applied Mathematics*, vol. 186, pp. 112–127, 2015.

[57] P. Segui-Gasco, H. Shin, A. Tsourdos, and V. Seguí, "Decentralized submodular multi-robot task allocation," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2015.

[58] R. Williams, A. Gasparri, and G. Ulivi, "Decentralized matroid optimization for topology constraints in multi-robot allocation problems," in *Proc. IEEE Conf. on Robotics and Automation*, 2017.

[59] M. Corah and N. Michael, "Efficient online multi-robot exploration via distributed sequential greedy assignment," in *Robotics: Science and Systems*, 2017.

[60] S. Jorgensen, R. Chen, M. Milam, and M. Pavone, "The matroid team surviving orienteers problem: Constrained routing of heterogeneous teams with risky traversal," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2017.

[61] ——, "The risk-sensitive coverage problem: Multi-robot routing under uncertainty with service level and survival constraints," in *Proc. IEEE Conf. on Decision and Control*, 2017.

[62] R. Iyer and J. Bilmes, "Submodular optimization with submodular cover and submodular knapsack constraints," in *Advances in Neural Information f Systems*, 2013.

[63] L. Gargano and M. Hammar, "A note on submodular set cover on matroids," *Discrete Mathematics*, vol. 309, no. 18, pp. 5739–5744, 2009.

[64] J. Ehmke, A. Campbell, and T. Urban, "Ensuring service levels in routing problems with time windows and stochastic travel times," *European Journal of Operational Research*, vol. 240, no. 2, pp. 539–550, 2015.

[65] Y. Adulyasak and P. Jaillet, "Models and algorithms for stochastic and robust vehicle routing with deadlines," *Transportation Science*, vol. 50, no. 2, pp. 608–626, 2015.

[66] E. Galceran and M. Carreras, "A survey on coverage and path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[67] S. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 410–426, 2012.

[68] A. Wallar, E. Plaku, and D. Sofge, "Reactive motion planning for unmanned aerial surveillance of risk-sensitive areas," *IEEE Transactions on Automation Sciences and Engineering*, vol. 12, no. 3, pp. 969–980, 2015.

[69] J.-F. Bérubé, M. Gendreau, and J.-Y. Potvin, "An exact ε-constraint method for bi-objective combinatorial optimization problems: Application to the Travelling Salesman Problem with Profits," *European Journal of Operational Research*, vol. 194, no. 1, pp. 39–50, 2009.

[70] C. Filippi and E. Stevanato, "Approximation schemes for bi-objective combinatorial optimization and their application to the TSP with profits," *Computers & Operations Research*, vol. 40, no. 10, pp. 2418–2428, 2013.

[71] ——, "A two-phase method for bi-objective combinatorial optimization and its application to the TSP with profits," *Algorithmic Operations Research*, vol. 7, no. 2, 2013.

[72] J. Yu, M. Schwager, and D. Rus, "Correlated orienteering problem and its application to persistent monitoring tasks," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1106–1118, 2016.

[73] G. Best, O. Cliff, T. Patten, R. Mettu, and R. Fitch, "Decentralised Monte Carlo tree search for active perception," in *Workshop on Algorithmic Foundations of Robotics*, 2016.

[74] A. Singh, A. Krause, and W. Kaiser, "Nonmyopic adaptive informative path planning for multiple robots," in *Int. Joint Conf. on Artificial Intelligence*, 2009.

[75] X.-H. Chen, A. Dempster, and J. Liu, "Weighted finite population sampling to maximize entropy," *Biometrika*, vol. 81, no. 3, pp. 457–469, 1994.